Pattern Recognition 47 (2014) 3060–3069

Contents lists available at ScienceDirect

Pattern Recognition

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

Cooperative and penalized competitive learning with application to kernel-based clustering

Hong Jia [a], Yiu-ming Cheung [a,b,*], Jiming Liu [a]

[a] Department of Computer Science, Hong Kong Baptist University, Hong Kong SAR, China
[b] The United International College, BNU-HKBU, Zhuhai, China

ARTICLE INFO

Article history:
Received 19 September 2013
Received in revised form
5 February 2014
Accepted 13 March 2014
Available online 21 March 2014

Keywords:
Competitive learning
Cooperation mechanism
Penalization mechanism
Kernel-based clustering
Number of clusters

ABSTRACT

Competitive learning approaches with individual penalization or cooperation mechanisms have the attractive ability of automatic cluster number selection in unsupervised data clustering. In this paper, we further study these two mechanisms and propose a novel learning algorithm called Cooperative and Penalized Competitive Learning (CPCL), which implements the cooperation and penalization mechanisms simultaneously in a single competitive learning process. The integration of these two different kinds of competition mechanisms enables the CPCL to locate the cluster centers more quickly and be insensitive to the number of seed points and their initial positions. Additionally, to handle nonlinearly separable clusters, we further introduce the proposed competition mechanism into kernel clustering framework. Correspondingly, a new kernel-based competitive learning algorithm which can conduct nonlinear partition without knowing the true cluster number is presented. The promising experimental results on real data sets demonstrate the superiority of the proposed methods.

1. Introduction

As an efficient approach to clustering analysis, competitive learning has been widely applied to a variety of research areas such as data mining [1], computer vision [2], bioinformatics [3] and so forth. However, in an unsupervised learning environment, clustering problems can be extremely difficult especially when the number of clusters is unavailable in advance. That is because traditional methods, including the *k*-means algorithm [4] and Expectation-Maximization (EM) algorithm [5], need the users to specify the exact number of clusters as an input; otherwise, they will likely produce incorrect clustering results. Generally, choosing the cluster number is an ad hoc decision based on prior knowledge of given data and it becomes nontrivial when the data has many dimensions [6].

In the literature, competitive learning with different mechanisms have received wide attention due to their effectiveness and interpretability for automatic cluster number detection. For example, with a penalization mechanism, the Rival Penalized Competitive Learning (RPCL) [7] can automatically select the cluster number by gradually driving extra seed points (i.e. the variables that are learnable towards the center of data clusters in the input space) far away from the dense region of the input data set. In this learning approach, for each data observation (also called *input*), not only the winner among all seed points is updated to adapt to the input, but also the second winner is penalized by a much smaller fixed rate (also called *delearning rate* hereinafter). However, empirical studies have found that the performance of RPCL algorithm is sensitive to the delearning rate, whose optimal setting differs for variant problems [7,8]. To solve this problem, an improved version named Rival Penalization Controlled Competitive Learning (RPCCL) [8] was proposed, which controls the rival-penalized strength through an adaptive way based on the distance between the winner and the rival relative to the current input. In general, as pointed out in [9], both of RPCL and RPCCL always penalize the extra seed points even if they are much far away from the dense region of the input data set. Consequently, the learning curves of seed points obtained by these algorithms as a whole will not tend to convergence. By contrast, another variant of RPCL called Stochastic RPCL (S-RPCL) [9], developed from the Rival Penalized Expectation-Maximization (RPEM) algorithm [9], can lead to a convergent learning process by penalizing the nearest rival stochastically based on its posterior probability. Moreover, to theoretically analyze the convergence behavior of rival penalized leaning method, Ma and Wang [10] have presented a general form of RPCL algorithm, called distance-sensitive RPCL (DSRPCL) and proved that the correct convergence of DSRPCL is associated with

* Corresponding author at: Department of Computer Science, Hong Kong Baptist University, Hong Kong SAR, China.
E-mail addresses: hongjia@comp.hkbu.edu.hk (H. Jia), ymc@comp.hkbu.edu.hk (Y.-m. Cheung), jiming@comp.hkbu.edu.hk (J. Liu).

http://dx.doi.org/10.1016/j.patcog.2014.03.010
0031-3203/

the minimization of a cost function defined on the weight vectors of a competitive learning network. It has also been pointed out in [10] that the DSRPCL algorithm may result in wrong convergence when the specified cluster number k becomes much larger than the true cluster number k^* (i.e. $k > 2k^*$). This phenomenon also exists in the other versions of the rival penalized learning algorithm as shown in [11]. In contrast, the Competitive Repetition Suppression (CoRe) clustering method proposed in [11] can give a good performance if it is initialized with sufficiently large number of seed points. This method is inspired by a cortical memory mechanism and extends the RPCL framework by allowing multiple winners and losers in each learning iteration.

Besides the penalization mechanism, a cooperation strategy can also be utilized for detecting the cluster number in the competitive learning paradigm. One example is the Competitive and Cooperative Learning (CCL) [12] algorithm, in which the winner of each learning iteration will dynamically cooperate with several nearest rivals to update towards the input data together. Consequently, the CCL can make all the seed points converge to the corresponding cluster centers and the number of those seed points stably locating at different positions is exactly the cluster number. Nevertheless, further empirical studies presented by Li et al. [13] indicate that the performance of CCL is somewhat sensitive to the initial positions of seed points. To overcome this difficulty, they have proposed an improved variant, namely Cooperation Controlled Competitive Learning (CCCL) method, in which the learning rate of each seed point within the same cooperative team is adjusted adaptively based on the distance between the cooperator and the current input. Nevertheless, some empirical studies have found that the CCCL algorithm may still not work well if the clusters are seriously overlapped or initial seed points are all gathered in one cluster.

To sum up, the competitive learning methods with a pure penalization or cooperation mechanism have different advantages and limitations. Therefore, designing a new competitive learning method, which features the merits of both penalization and cooperation mechanisms while counteracts their respective drawbacks, will definitely improve the accuracy of clustering analysis without knowing cluster number. However, since these two kinds of competitive mechanisms conduct an opposite learning process, i.e. penalization is to drive extra seed points far away from the input space while cooperation is to converge all seed points to the corresponding cluster centers, it is a nontrivial task to combine them into a single learning procedure. Moreover, all of the aforementioned competitive learning methods are based on the framework of k-means approach and only suitable for linearly separable clusters. Nevertheless, a nonlinearly separable cluster structure is common from a practical viewpoint. In the literature, kernel-based clustering methods have been widely used to analyze nonlinear clusters [14,15]. This kind of approaches utilizes kernel functions to map the original data into a high dimensional feature space, in which a linear partition will result in a nonlinear partition in the input space. Corresponding algorithms include the kernel k-means [16], global kernel k-means [17], kernel SOM [18,19] and so on. However, all these kernel clustering algorithms also need the number of clusters to be specified exactly, which becomes difficult in an unsupervised learning environment. To the best of our knowledge, conducting kernel-based clustering without knowing the cluster number has not been well studied yet.

In this paper, we further study the penalization and cooperation mechanisms, which actually conduct opposite shift tracks on the seed points (i.e. scatteration and aggregation), and explore a novel learning model which can simultaneously inherit the advantages of these two different kinds of mechanisms. Specifically, a new competitive learning algorithm, namely Cooperative and Penalized Competitive Learning (CPCL), which performs cooperation and penalization in a single competitive learning process is presented. In this method, given an input, the winner generated from the competition of all seed points will not only dynamically select several nearest competitors to form a cooperative team to adapt to the input together, but also penalize some other seed points which compete intensively with it. The cooperation mechanism here enables the closest seed points to update together and gradually converge to the corresponding cluster centers while the penalization mechanism supplies the other seed points with the opportunity to wander in the clustering space and search for more appropriate cluster centers. Consequently, this algorithm features the fast convergence speed and the robust performance against the initialization of seed points. Moreover, to handle the nonlinearly separable clusters, we further introduce the proposed cooperative and penalized competitive mechanism into the learning framework of the kernel k-means method. Correspondingly, a new kernel-based clustering algorithm which can nonlinearly partition given data without knowing the true cluster number is presented. Experiments on variant real data sets have demonstrated the good performance of proposed algorithms.

The rest of this paper is organized as follows. Section 2 describes the proposed CPCL approach and gives out the corresponding algorithm. Section 3 introduces the CPCL method into the kernel k-means framework and presents a new algorithm to solve the cluster number selection problem in kernel-based clustering. Then, Section 4 shows the experimental results on various real data sets. Finally, we draw a conclusion in Section 5.

2. Cooperative and penalized competitive learning (CPCL) approach

To simultaneously inherit the advantages of the two opposite competitive strategies (i.e., cooperation and penalization), we present a novel competitive learning model namely Cooperative and Penalized Competitive Learning (CPCL), which can perform cooperation and penalization in a single competitive learning process.

2.1. Cooperation and Penalization Mechanisms in CPCL

This sub-section describes the cooperation and penalization mechanisms of CPCL approach. Suppose we have N inputs, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, coming from k^* unknown clusters, and k ($k \geq k^*$) seed points, $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_k$, which are randomly initialized in the input space. Subsequently, given an input $\mathbf{x}_t$ each time, the winner among k seed points denoted as $\mathbf{m}_c$ is determined by

$$I(j|\mathbf{x}_t) = \begin{cases} 1 & \text{if } j = c = \arg\min\limits_{1 \leq i \leq k} \{\gamma_i \| \mathbf{x}_t - \mathbf{m}_i \|^2\} \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

with the relative winning frequency γ_i of $\mathbf{m}_i$ defined as

$$\gamma_i = \frac{n_i}{\sum_{j=1}^{k} n_j}, \tag{2}$$

where n_i is the winning times of $\mathbf{m}_i$ in the past [20].

After selecting out the winner $\mathbf{m}_c$, the area centered at $\mathbf{m}_c$ with the radius $\| \mathbf{m}_c - \mathbf{x}_t \|$ is regarded as the territory of $\mathbf{m}_c$. Fig. 1 has shown the winner's territory in two-dimensional space. Any other seed points which have intruded into this territory will be dominated by $\mathbf{m}_c$. That is, any other seed point $\mathbf{m}_j$ which satisfies

$$\| \mathbf{m}_c - \mathbf{m}_j \| \leq \| \mathbf{m}_c - \mathbf{x}_t \| \tag{3}$$

will either cooperate with the winner or be penalized by it, such as $\mathbf{m}_1$, $\mathbf{m}_2$ and $\mathbf{m}_3$ in Fig. 1. Subsequently, a question is naturally arisen: how to design the cooperation and penalization mechanism for a seed point?

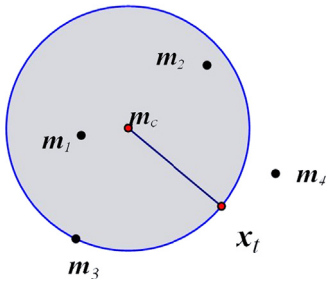

Fig. 1. The territory of the winner $\mathbf{m}_c$, indicated by a shadow circle, in the competition with the other seed points as given an input $\mathbf{x}_t$.

Before showing the cooperation mechanism, let us consider the cooperation behavior in our social life first. Usually, we can find that a person needs to evaluate its competitor's reliability before setting up cooperation relationship. If the reliability is high, it will be content to cooperate with this competitor to gain more benefits; otherwise, the cooperative relationship will not be built. Inspired by this phenomenon, in CPCL model, we assign a confidence coefficient, denoted as E_c, to the winning seed point $\mathbf{m}_c$. Here, E_c is a value fallen into the range [0,1] and measures the reliability of a winner. As long as E_c is calculated, the cooperation region of $\mathbf{m}_c$ can be determined by its confidence level. That is, the more reliable the winner is, the more seed points in its territory will cooperate with it. Then, the subsequent question is how to compute E_c?

It is known that the reliability of an individual is influenced by its experience. Evidently, more successful experience results in higher reliability. Similarly, the confidence level of a winning seed point in clustering learning should be changed over its learning experience. Specifically, we let R_c be the accumulated reliability of the winner $\mathbf{m}_c$ with

$$R_c = \eta \cdot n_c, \tag{4}$$

where η is a pre-specified small positive learning rate and n_c denotes the winning times of $\mathbf{m}_c$ in the past. That is, the reliability of a winning seed point is determined by its winning times and learning extent of each time. It can be seen that the more times a seed point has won in the past, the more reliability it will accumulate. Consequently, the confidence coefficient E_c of the winner $\mathbf{m}_c$ can be given by

$$E_c = \min(1, R_c) = \min(1, \eta \cdot n_c). \tag{5}$$

That is, the value of E_c will gradually increase over R_c and the maximum is 1, which means a full confidence. Subsequently, the number of cooperators owned by a winner can be determined by its confidence coefficient E_c.

Suppose there are q seed points which have intruded into the winner's territory, then the number of cooperators q_u can be calculated by

$$q_u = \lfloor q \cdot E_c \rfloor = \lfloor q \cdot \min(1, \eta \cdot n_c) \rfloor, \tag{6}$$

where $\lfloor \cdot \rfloor$ denotes the floor function. It implies that the number of cooperators will never exceed the number of interlopers in the winner's territory. Furthermore, if E_c is equal to 0, no seed point will cooperate with the winner any more because it has no reliability.

In the proposed learning model, the competitor nearest to the winning seed point has the priority to be a cooperator of the winner. Specifically, if q_u is equal to 1, the seed point that is the nearest to the winner will cooperate with the winner. Additionally, if q_u is equal to 2, the top two competitors with the smallest distance to the winner will be chosen as the cooperators. Subsequently, all of the other non-cooperating intruders in the winner's territory will be penalized. Hence, the number of penalized seed points, denoted as q_p, is determined by

$$q_p = q - q_u = q - \lfloor q \cdot \min(1, \eta \cdot n_c) \rfloor = \lceil q \cdot \max(0, 1 - \eta \cdot n_c) \rceil, \tag{7}$$

where $\lceil \cdot \rceil$ means the ceiling function. Evidently, the winning times of each seed point are very few at the initial stage, i.e. $q_u = 0$ and $q_p = q$. As a result, all the seed points which have fallen into the winner's territory will be penalized. Later, the power of cooperation will escalate over time, meanwhile the penalization region will be gradually reduced. In other words, at the earlier stage of CPCL model, the penalization mechanism plays a leading role, which makes the initial seed points drift in the input space to find a more appropriate cluster center. Later, the cooperation gradually becomes strengthened over time while the penalization is weakened. This makes all the seed points converge to the corresponding cluster centers gradually.

After determining the cooperating team and penalized team at time t, each cooperator, denoted as $\mathbf{m}_u$, will be updated by

$$\mathbf{m}_u^{(t)} = \mathbf{m}_u^{(t-1)} + \eta \rho_u (\mathbf{x}_t - \mathbf{m}_u^{(t-1)}), \tag{8}$$

where

$$\rho_u = \frac{\| \mathbf{m}_c^{(t-1)} - \mathbf{x}_t \|}{\max(\| \mathbf{m}_c^{(t-1)} - \mathbf{x}_t \|, \| \mathbf{m}_u^{(t-1)} - \mathbf{x}_t \|)}. \tag{9}$$

It can be seen that all the cooperative seed points are updated with a movement towards the point $\mathbf{x}_t$, but the strength of their adjustment is individually based on the distance between a cooperator and the current input $\mathbf{x}_t$. Since we have a frequency factor γ involving in Eq. (1) when selecting the winning seed, the nearest seed point to $\mathbf{x}_t$ is not always the winner. Therefore, the "max" function in Eq. (8) is needed. We can find that a cooperator will have a full learning strength as $\| \mathbf{m}_u^{(t-1)} - \mathbf{x}_t \| \le \| \mathbf{m}_c^{(t-1)} - \mathbf{x}_t \|$, i.e. the ratio ρ in Eq. (8) is equal to 1. Otherwise, the learning strength is gradually reduced over the distance between a cooperator and $\mathbf{x}_t$.

Meanwhile, the other penalized seed points in the winner's territory, denoted as $\mathbf{m}_p$, will be penalized with a dynamical penalizing rate:

$$\mathbf{m}_p^{(t)} = \mathbf{m}_p^{(t-1)} - \eta \frac{\| \mathbf{m}_c^{(t-1)} - \mathbf{x}_t \|}{\| \mathbf{m}_p^{(t-1)} - \mathbf{x}_t \|} (\mathbf{x}_t - \mathbf{m}_p^{(t-1)}). \tag{10}$$

That is, all the penalized seed points will be moved away from $\mathbf{x}_t$. The closer the seed point is to $\mathbf{x}_t$, the more penalization it will suffer from.

2.2. The CPCL algorithm

Following the description of Section 2.1, the CPCL algorithm for data clustering can be summarized as Algorithm 1. In the algorithm, e stands for the update of all seed points between two consecutive iterations, which is calculated by

$$e = \sum_{j=1}^{k} \| \mathbf{m}_j^{(T)} - \mathbf{m}_j^{(T-1)} \|^2. \tag{11}$$

ε is a very small number and $e \le \varepsilon$ indicates the convergency of all seed points. T stands for the current number of learning epochs, i.e. the times that the whole data set has been scanned. During our experiments, the value of ε has been set at 10^{-5}.

Algorithm 1. Cooperative and Penalized Competitive Learning (CPCL).

1: **Input**: data set X, learning rate η and an initial value of k ($k \ge k^*$)
2: **Output**: cluster label $Y = \{y_1, y_2, \ldots, y_N\}$ and cluster number k^*

3: Randomly initialize the k seed points, denoted as $\{\mathbf{m}_1^{(0)}, \mathbf{m}_2^{(0)}, \ldots, \mathbf{m}_k^{(0)}\}$. Set $n_j^{(0)} = 1$ with $j = 1, 2, \ldots, k$, and $t = 1$.
4: **repeat**
5: **for** $i = 1$ **to** N **do**
6: Calculate $I(j|\mathbf{x}_i)$ by Eq. (1) and determine the winning unit $\mathbf{m}_c^{(t-1)}$.
7: Let S_c be the set of seed points fallen into the territory of $\mathbf{m}_c^{(t-1)}$. Set $S_c = \varnothing$, and then we add $\mathbf{m}_j^{(t-1)}$ ($j \in \{1, 2, \ldots, k\}$, $j \neq c$) into S_c if it satisfies Eq. (3).
8: Sort the units in S_c as $\{\mathbf{m}_1'^{(t-1)}, \mathbf{m}_2'^{(t-1)}, \ldots, \mathbf{m}_q'^{(t-1)}\}$ such that $\|\mathbf{m}_1'^{(t-1)} - \mathbf{m}_c^{(t-1)}\| \leq \|\mathbf{m}_2'^{(t-1)} - \mathbf{m}_c^{(t-1)}\| \leq \cdots \leq \|\mathbf{m}_q'^{(t-1)} - \mathbf{m}_c^{(t-1)}\|$, where $q = |S_c|$.
9: Select a subset S_u of S_c to form a cooperating team of $\mathbf{m}_c^{(t-1)}$, where $S_u = \{\mathbf{m}_1'^{(t-1)}, \mathbf{m}_2'^{(t-1)}, \ldots, \mathbf{m}_{q_u}'^{(t-1)}\}$ and q_u is calculated by Eq. (6). Then update all members in S_u by Eq. (8).
10: Let $S_p = S_c - S_u$, then penalize all seed points in S_p by Eq. (10).
11: Update the winner $\mathbf{m}_c$ by
$$\mathbf{m}_c^{(t)} = \mathbf{m}_c^{(t-1)} + \eta \cdot (\mathbf{x}_i - \mathbf{m}_c^{(t-1)}). \quad (12)$$
12: Update n_c by $n_c^{(t)} = n_c^{(t-1)} + 1$, and increase t by 1.
13: **end for**
14: **until** $e \leq \varepsilon$ or $T \geq T_{max}$

To illustrate the learning process of CPCL algorithm, we have applied it to a set of data from a mixture of three Gaussian densities as shown in Fig. 2. Five seed points were initialized in the input space and their positions are marked by ○ in the figure. After the convergency of CPCL, the three clusters were identified and the learned positions of cluster centers have been marked by ⋆ for visualization. Moreover, the trajectories of seed points during the training process have been depicted with the solid lines in Fig. 2, which illustrate how the true cluster number is learned. Additionally, a snapshot of the convergent positions of five seed points is as follows:

$\mathbf{m}_1 = (1.0286, 0.9578)^T$, $\mathbf{m}_2 = (0.9791, 2.4587)^T$,
$\mathbf{m}_3 = (0.9791, 2.4587)^T$, $\mathbf{m}_4 = (2.5180, 2.4980)^T$,
$\mathbf{m}5 = (2.5180, 2.4980)^T$.

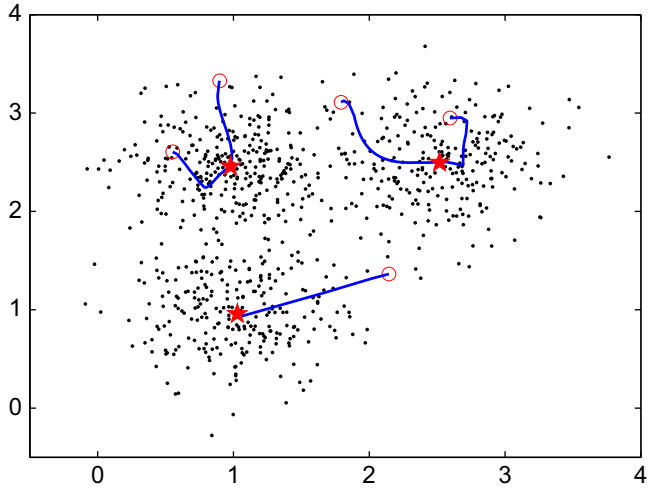

Fig. 2. An illustration of the learning process of CPCL algorithm on the data coming from a mixture of three Gaussian densities, where '○' marks the initial positions of seed points, '⋆' represents the cluster centers learned by CPCL, and the solid lines depict the trajectories of seed points.

Evidently, $\mathbf{m}_2$ and $\mathbf{m}_3$ have converged to the same cluster center, meanwhile, $\mathbf{m}_4$ and $\mathbf{m}_5$ have overlapped in another cluster center. Therefore, based on the learning result of CPCL algorithm, the number of seed points with different positions is exactly the cluster number.

2.3. Comparisons between CPCL and existing counterparts

As described in Section 2, the proposed CPCL method synchronously implements two different kinds of competitive mechanisms: cooperation and penalization, which have been individually adopted by some existing methods. Therefore, in this part, we will analyze the characteristic of CPCL approach and discuss the similarities and differences between it and other existing counterparts.

The key common ground between CPCL and the cooperative learning methods, i.e. CCL [12] and CCCL [13] algorithms, is that they both converge all seed points within the input space to the corresponding cluster centers without pushing anyone away. It can be seen that the CPCL will degenerate to the CCCL method with the parameter $\varphi = 1$ if the winning seed point $\mathbf{m}_c$ is always full reliable at any time, i.e. $E_c = 1$ and $q_u = q$. Nevertheless, the penalization mechanism in CPCL generally offers the seed points more opportunities to wander in the clustering space and search for a more appropriate cluster center. As a result, the CPCL is more insensitive to the overlapping level of clusters and the initialization of the seed points in comparison with the purely cooperative learning methods.

Compared to the existing penalized learning methods, such as RPCL [7] and DSRPCL [10], the proposed CPCL method has implemented the penalization strategy in a different way. Specifically, the penalized seed points in CPCL are the ones that locate close to the winner and have fallen into the defined winner's territory, but not the second winner that adopted in the rival penalized learning methods. This is because the goal of penalizing operation in CPCL is to disperse the seed points located around the winner and let them have a sufficient exploration in the clustering space during the early stage of the learning process. By contrast, the penalization in other existing methods is to drive the redundant seed points away from the input space. Under this new strategy, if there is a seed point which is very close to the input sample but has not fallen into current winner's territory, it will not be penalized but has the opportunity to be a winner during the following learning procedures in that the winning chance of each seed point is restrained by its winning times. Moreover, in the CPCL algorithm, the penalization mechanism will work only in the beginning period and be gradually faded out over time. Hence, in the anaphase, the trajectories of all seed points will have a good convergent behavior under the domination of cooperation strategy.

Besides, another penalized learning model in the literature called CoRe [11] also allows multiple winners and losers during each learning iteration. However, the competitive strategies and learning process of CoRe and CPCL are totally different. In each iteration of CoRe algorithm, all the seed points are partitioned into winners and losers based on the similarities between them and the given data point with a predefined threshold. During the implementation of CoRe, this threshold is suggested to be set at 0.9. Under this constraint, in most cases only one seed point will be assigned to the winner set, and all the other seed points are penalized. By contrast, the CPCL determines the cooperators and penalized losers in a self-adapting way and all the seed points have three kinds of situations in each iteration: cooperative, penalized, and unaltered. Furthermore, since CoRe has large penalization range and strength, it needs a sufficiently large value for initial k. By contrast, the proposed CPCL is effective with $k \geq k^*$.

Moreover, since the winning seed point in the CPCL algorithm chooses some cooperators meanwhile penalizes the other competitors,

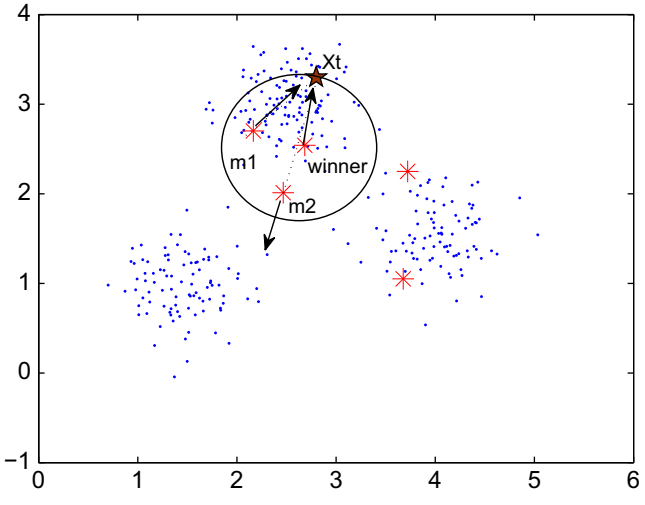

Fig. 3. Simultaneous cooperation and penalization in the CPCL approach.

this method is expected to have a good convergence speed. For example, as shown in Fig. 3, the seed point $\mathbf{m}_1$ is chosen as a cooperator while $\mathbf{m}_2$ is penalized. It can be observed that, while the winner and $\mathbf{m}_1$ are adjusted adaptively to $\mathbf{x}_t$, $\mathbf{m}_2$ is pushed away from this cluster area and moves towards another cluster center. The synergy of these two different kinds of moving trends can make the seed points converge to the corresponding cluster centers more quickly. The numerical results in the experiment section will show the promising characteristic of the CPCL method.

3. CPCL for kernel-based clustering

To solve the cluster number selection problem in kernel-based clustering, this section introduces the cooperative and penalized competitive learning mechanism into the online version of kernel k-means clustering framework. Subsequently, a novel clustering algorithm which can conduct a nonlinear partition on the input data set without knowing exact number of clusters will be presented.

3.1. Kernel-based competitive learning

Given the data set $X=\{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_N\}$ with $\mathbf{x}_i\in\mathbb{R}^d$, the Mercer kernel $K: X\times X\to\mathbb{R}$ can be expressed as

$$K(\mathbf{x}_i,\mathbf{x}_j)=\Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j),\quad \forall i,\ j\in\{1,2,\ldots,N\}. \tag{13}$$

Here, $\Phi: X\to\mathcal{F}$ maps the original space X to a high dimensional feature space $\mathcal{F}$. The clustering in feature space is to find k centers (i.e., $\mathbf{m}_j^{\Phi}\in\mathcal{F}$ with $j=1,2,\ldots,k$), which partition the mapped patterns into different groups such that the summation of distances between each center and its cluster members in feature space is minimized. Generally, each center $\mathbf{m}_j^{\Phi}$ can be written as a combination of the mapped patterns [16]. Accordingly, we have

$$\mathbf{m}_j^{\Phi}=\sum_{i=1}^{N}\alpha_{ji}\Phi(\mathbf{x}_i), \tag{14}$$

where α_{ji} is a non-negative coefficient. Subsequently, based on the kernel trick [16,21], the squared distance between a mapped pattern $\Phi(\mathbf{x}_i)$ and a center $\mathbf{m}_j^{\Phi}$ can be calculated by

$$\begin{aligned}\|\Phi(\mathbf{x}_i)-\mathbf{m}_j^{\Phi}\|^2&=\|\Phi(\mathbf{x}_i)-\sum_{t=1}^{N}\alpha_{jt}\Phi(\mathbf{x}_t)\|^2\\&=K(\mathbf{x}_i,\mathbf{x}_i)-2\sum_{t=1}^{N}\alpha_{jt}K(\mathbf{x}_i,\mathbf{x}_t)+\sum_{r,s=1}^{N}\alpha_{jr}\alpha_{js}K(\mathbf{x}_r,\mathbf{x}_s).\end{aligned} \tag{15}$$

For the competitive learning method, given a data point $\mathbf{x}_t$ each time, the winner $\mathbf{m}_c^{\Phi}$ among k centers is determined by

$$c=\arg\min_{1\le j\le k}\{\gamma_j\|\Phi(\mathbf{x}_t)-\mathbf{m}_j^{\Phi}\|^2\}, \tag{16}$$

where γ_j is the relative winning frequency of $\mathbf{m}_j^{\Phi}$. Synthesizing Eqs. (15) and (16), we can get

$$c=\arg\min_{1\le j\le k}\left\{\gamma_j\left[\sum_{r,s=1}^{N}\alpha_{jr}\alpha_{js}K(\mathbf{x}_r,\mathbf{x}_s)-2\sum_{i=1}^{N}\alpha_{ji}K(\mathbf{x}_t,\mathbf{x}_i)\right]\right\}. \tag{17}$$

Subsequently, $\mathbf{x}_t$ is assigned to the winning cluster and the corresponding cluster center is updated with

$$\mathbf{m}_c^{\Phi(t)}=\mathbf{m}_c^{\Phi(t-1)}+\eta(\Phi(\mathbf{x}_t)-\mathbf{m}_c^{\Phi(t-1)}), \tag{18}$$

where η is a small learning rate. Substituting Eq. (14) into Eq. (18) yields

$$\begin{aligned}\sum_{i=1}^{N}\alpha_{ci}^{(t)}\Phi(\mathbf{x}_i)&=\sum_{i=1}^{N}\alpha_{ci}^{(t-1)}\Phi(\mathbf{x}_i)+\eta\Phi(\mathbf{x}_t)-\eta\sum_{i=1}^{N}\alpha_{ci}^{(t-1)}\Phi(\mathbf{x}_i)\\&=(1-\eta)\sum_{i=1}^{N}\alpha_{ci}^{(t-1)}\Phi(\mathbf{x}_i)+\eta\Phi(\mathbf{x}_t).\end{aligned} \tag{19}$$

Therefore, the updating of winning center $\mathbf{m}_c^{\Phi}$ can be handled indirectly by updating the coefficient α_{ci} according to

$$\alpha_{ci}^{(t)}=\begin{cases}(1-\eta)\alpha_{ci}^{(t-1)} & \text{if } i\ne t,\\ (1-\eta)\alpha_{ci}^{(t-1)}+\eta & \text{otherwise.}\end{cases} \tag{20}$$

3.2. Implementation of cooperation and penalization mechanisms

To automatically learn the true number of clusters, we introduce the cooperation and penalization mechanisms presented above into the competitive learning framework and propose a new algorithm which can conduct kernel-based clustering without knowing exact cluster number. Specifically, once the winner $\mathbf{m}_c^{\Phi}$ is selected, the other cluster centers which have fallen into its territory will be dominated by it. That is, any center $\mathbf{m}_j^{\Phi}$ $(j\ne c)$ satisfies

$$\|\mathbf{m}_c^{\Phi}-\mathbf{m}_j^{\Phi}\|^2\le\|\mathbf{m}_c^{\Phi}-\Phi(\mathbf{x}_t)\|^2 \tag{21}$$

will cooperate with the winner or be penalized by it. Based on Eqs. (14) and (15), Eq. (21) can be rewritten as

$$\sum_{r,s=1}^{N}(\alpha_{jr}\alpha_{js}-2\alpha_{cr}\alpha_{js})K(\mathbf{x}_r,\mathbf{x}_s)\le K(\mathbf{x}_t,\mathbf{x}_t)-2\sum_{i=1}^{N}\alpha_{ci}K(\mathbf{x}_t,\mathbf{x}_i). \tag{22}$$

Let set S_c^{Φ} store all the cluster centers which satisfy Eq. (22). Suppose there are q items in S_c^{Φ}, then it can be represented as $S_c^{\Phi}=\{\mathbf{m}_1^{\Phi},\mathbf{m}_2^{\Phi},\ldots,\mathbf{m}_q^{\Phi}\}$. To select the cooperating team, we first sort the cluster centers in S_c^{Φ} according to their respective distance to the winner $\mathbf{m}_c^{\Phi}$. When compare the distance, we have

$$\begin{aligned}&\|\mathbf{m}_c^{\Phi}-\mathbf{m}_1^{\Phi}\|^2\le\|\mathbf{m}_c^{\Phi}-\mathbf{m}_2^{\Phi}\|^2\\&\Leftrightarrow\|\sum_{i=1}^{N}\alpha_{ci}\Phi(\mathbf{x}_i)-\sum_{i=1}^{N}\alpha_{1i}\Phi(\mathbf{x}_i)\|^2\le\|\sum_{i=1}^{N}\alpha_{ci}\Phi(\mathbf{x}_i)-\sum_{i=1}^{N}\alpha_{2i}\Phi(\mathbf{x}_i)\|^2\\&\Leftrightarrow\sum_{r,s=1}^{N}(\alpha_{1r}\alpha_{1s}-2\alpha_{cr}\alpha_{1s})K(\mathbf{x}_r,\mathbf{x}_s)\\&\le\sum_{r,s=1}^{N}(\alpha_{2r}\alpha_{2s}-2\alpha_{cr}\alpha_{2s})K(\mathbf{x}_r,\mathbf{x}_s).\end{aligned} \tag{23}$$

Let $\Delta_{cj}=\sum_{r,s=1}^{N}(\alpha_{jr}\alpha_{js}-2\alpha_{cr}\alpha_{js})K(\mathbf{x}_r,\mathbf{x}_s)$, then, the q units in S_c^{Φ} can be sorted as $\{\mathbf{m}_{1'}^{\Phi},\mathbf{m}_{2'}^{\Phi},\ldots,\mathbf{m}_{q'}^{\Phi}\}$ such that

$$\Delta_{c1'}\le\Delta_{c2'}\le\ldots\le\Delta_{cq'}. \tag{24}$$

Subsequently, the cooperating cluster centers are selected in order from the sorted set and the number of cooperators is determined by Eq. (6). All the cooperators will be adjusted towards the given data point with dynamic learning rate according to

$$\mathbf{m}_u^{\Phi(t)}=\mathbf{m}_u^{\Phi(t-1)}+\eta\rho_u(\Phi(\mathbf{x}_t)-\mathbf{m}_u^{\Phi(t-1)}), \tag{25}$$

where

$$\rho_u = \frac{\| \mathbf{m}_c^{\Phi(t-1)} - \Phi(\mathbf{x}_t) \|^2}{\max(\| \mathbf{m}_c^{\Phi(t-1)} - \Phi(\mathbf{x}_t) \|^2, \| \mathbf{m}_u^{\Phi(t-1)} - \Phi(\mathbf{x}_t) \|^2)}. \quad (26)$$

Based on Eq. (14), Eq. (25) can be further rewritten as

$$\alpha_{ui}^{(t)} = \begin{cases} (1-\eta\rho_u)\alpha_{ui}^{(t-1)} & \text{if } i \neq t, \\ (1-\eta\rho_u)\alpha_{ui}^{(t-1)} + \eta\rho_u & \text{otherwise.} \end{cases} \quad (27)$$

The other non-cooperating centers in S_c^{Φ} will be penalized by the winner. Correspondingly, the updating formula is given by

$$\mathbf{m}_p^{\Phi(t)} = \mathbf{m}_p^{\Phi(t-1)} - \eta\rho_p(\Phi(\mathbf{x}_t) - \mathbf{m}_p^{\Phi(t-1)}), \quad (28)$$

where

$$\rho_p = \frac{\| \mathbf{m}_c^{\Phi(t-1)} - \Phi(\mathbf{x}_t) \|^2}{\| \mathbf{m}_p^{\Phi(t-1)} - \Phi(\mathbf{x}_t) |^2}. \quad (29)$$

Substituting Eq. (14) into Eq. (28), we can get

$$\begin{aligned} \sum_{i=1}^{N} \alpha_{pi}^{(t)} \Phi(\mathbf{x}_i) &= \sum_{i=1}^{N} \alpha_{pi}^{(t-1)} \Phi(\mathbf{x}_i) - \eta\rho_p \Phi(\mathbf{x}_t) + \eta\rho_p \sum_{i=1}^{N} \alpha_{pi}^{(t-1)} \Phi(\mathbf{x}_i) \\ &= (1+\eta\rho_p) \sum_{i=1}^{N} \alpha_{pi}^{(t-1)} \Phi(\mathbf{x}_i) - \eta\rho_p \Phi(\mathbf{x}_t). \end{aligned} \quad (30)$$

Therefore, the updating of $\mathbf{m}_p^{\Phi}$ expressed by Eq. (28) is equivalent to

$$\alpha_{pi}^{(t)} = \begin{cases} (1+\eta\rho_p)\alpha_{pi}^{(t-1)} & \text{if } i \neq t, \\ (1+\eta\rho_p)\alpha_{pi}^{(t-1)} - \eta\rho_p & \text{otherwise.} \end{cases} \quad (31)$$

3.3. Kernel-based CPCL algorithm

Based on the description given in the former sub-sections, the cooperative and penalized competitive learning method for kernel-based clustering analysis can be summarized as Algorithm 2. Specially, to randomly initialize the k cluster centers in feature space, we make a random permutation on the order of input data and then initialize the centers as the first k mapped patterns. That is, we set $\alpha_{ji} = \delta_{ji}$, where $\delta_{ji} = 1$ if $i=j$ and 0 otherwise. The convergency index e^{Φ} is calculated by

$$\begin{aligned} e^{\Phi} &= \sum_{j=1}^{k} \| \mathbf{m}_j^{\Phi(T)} - \mathbf{m}_j^{\Phi(T-1)} \|^2 \\ &= \sum_{j=1}^{k} \| \sum_{i=1}^{N} \alpha_{ji}^{(T)} \Phi(\mathbf{x}_i) - \sum_{i=1}^{N} \alpha_{ji}^{(T-1)} \Phi(\mathbf{x}_i) \|^2 \\ &= \sum_{j=1}^{k} \left[\sum_{r,s=1}^{N} \alpha_{jr}^{(T)} \alpha_{js}^{(T)} K(\mathbf{x}_r, \mathbf{x}_s) - 2 \sum_{r,s=1}^{N} \alpha_{jr}^{(T)} \alpha_{js}^{(T-1)} K(\mathbf{x}_r, \mathbf{x}_s) \right. \\ &\quad \left. + \sum_{r,s=1}^{N} \alpha_{jr}^{(T-1)} \alpha_{js}^{(T-1)} K(\mathbf{x}_r, \mathbf{x}_s) \right], \end{aligned} \quad (32)$$

where $T-1$ and T stand for two sequential learning epochs. Moreover, in our experimental studies, Gaussian kernel has been utilized. Therefore, we have

$$K(\mathbf{x}_r, \mathbf{x}_s) = \exp\left(-\frac{\| \mathbf{x}_r - \mathbf{x}_s \|^2}{2\sigma^2} \right), \quad (33)$$

where σ is a suitable constant.

Algorithm 2. Kernel-based CPCL Algorithm (K-CPCL).

1: **Input**: data set X, learning rate η and an initial value of k ($k \geq k^*$)
2: **Output**: cluster label $Y = \{y_1, y_2, \ldots, y_N\}$ and cluster number k^*
3: Randomly initialize the k cluster centers, denoted as $\{\mathbf{m}_1^{\Phi(0)}, \mathbf{m}_2^{\Phi(0)}, \ldots, \mathbf{m}_k^{\Phi(0)}\}$. Set $n_j^{(0)} = 1$ with $j = 1, 2, \ldots, k$, and $t = 1$.
4: **repeat**
5: **for** $i = 1$ **to** N **do**
6: Determine the winning unit $\mathbf{m}_c^{\Phi(t-1)}$ according to Eq. (17).
7: Let $S_c^{\Phi} = \varnothing$, and then add $\mathbf{m}_j^{\Phi(t-1)}$ ($j \in \{1, 2, \ldots, k\}, j \neq c$) into S_c^{Φ} if it satisfies Eq. (22).
8: Sort the units in S_c^{Φ} according to Eq. (24).
9: Select a subset S_u^{Φ} of S_c^{Φ} to form a cooperating team and update all members in S_u^{Φ} by Eq. (27).
10: Let $S_p^{\Phi} = S_c^{\Phi} - S_u^{\Phi}$, then penalize all centers in S_p^{Φ} by Eq. (31).
11: Update the winner $\mathbf{m}_c^{\Phi}$ by Eq. (20).
12: Update n_c by $n_c^{(t)} = n_c^{(t-1)} + 1$, and increase t by 1.
13: **end for**
14: **until** $e^{\Phi} \leq \varepsilon$ or $T \geq T_{max}$

4. Experimental results

In this section, we present the experimental results of the proposed cooperative and penalized competitive learning method on real data sets in comparison with some existing counterparts. Each algorithm in the experiments was coded with MATLAB and all experiments were implemented by a desktop PC computer with Intel(R) Core(TM)2 Quad CPU, 2.40 GHz main frequency, and 4GB DDR2 667 RAM.

Since the true labels of tested data sets in our experiments are available, according to [6], the performance of clustering algorithms with capability of cluster number selection can be evaluated by following Partition Quality (PQ) index:

$$PQ = \begin{cases} \frac{\sum_{i=1}^{k^*} \sum_{j=1}^{k'} [p(i,j)^2 \cdot (p(i,j)/p(j))]}{\sum_{i=1}^{k^*} p(i)^2} & \text{if } k' > 1, \\ 0 & \text{otherwise,} \end{cases} \quad (34)$$

where k^* is the true number of classes and k' is the cluster number learned by the algorithm. The term $p(i,j)$ calculates the frequency-based probability that a data point is labeled i by the true label and labeled j by the obtained label. This PQ metric achieves the maximum value 1 when the obtained labels induce the same partition as the true ones. That is, all data points in each cluster have the same true label and the estimated k' is equal to k^*. Additionally, we have also utilized the Rand Index (RI) to measure the clustering accuracy for reference, which is given by

$$\text{RI} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}, \quad (35)$$

where TP, TN, FP, and FN stand for true positive, true negative, false positive, and false negative, respectively.

4.1. Performance of CPCL algorithm

4.1.1. Experiments on low dimensional data

This sub-section demonstrates the effectiveness of CPCL algorithm on three data sets: *Seeds*, *Wine*, and *Wisconsin Diagnostic Breast Cancer* (WDBC), with relatively low dimensionality obtained from UCI Machine Learning Data Repository.[1] For comparative studies, the results of CPCL algorithm have been compared with four existing methods, which are DSRPCL [10], RPCCL [8], CoRe [11], and CCCL [13] algorithms. These competitive learning algorithms have been executed

[1] http://archive.ics.uci.edu/ml/

with different settings of k and repeated 20 times in each case on every data set. The learning rate η of CPCL, RPCCL, and CCCL algorithms was set at 0.001 and in the CCCL method, the parameter φ was set at 0.5 according to [13]. For the RPCCL algorithm, we assigned a large number of learning epochs to let the extra seed points be sufficiently penalized. Moreover, for the CoRe method, we have selected a relatively good value for the most important parameter α from different trials in each experiment, and the settings of other parameters are consistent with the author's instructions.

In the dataset of Seeds, it has 210 instances with 7 attributes. All the instances are equally distributed into three varieties of wheat: Kama, Rosa and Canadian. Table 1 records the average values obtained by different clustering algorithms under variant evaluation criteria, including number of clusters, Partition Quality, Rand Index, running time, and number of learning epochs. From the results, we can find that CPCL has given a good estimate for the cluster number and obtained the best partition quality in each testing case. By contrast, the RPCCL method performed well when the initial k was slightly larger than the true one, i.e. $k=4$, but tended to be dragged into incorrect convergence when k was set much larger than k^*. It also can be observed that the partition quality of CoRe algorithm improved as the value of initial k increased. This phenomenon verifies that CoRe method needs a sufficiently large k to achieve a satisfying result. However, larger k also means heavier computation as shown in the statistic information of running time and epoch numbers.

In the dataset of Wine, it contains 178 instances from 3 types of wines. Each instance is described by 13 attributes. The numbers of instances in the three true classes are 59, 71, and 48, respectively. The clustering results obtained by different algorithms under variant settings have been summarized in Table 2. Referring to the obtained average value of cluster numbers, we can find that only the CPCL algorithm has given out an accurate estimate on the number of clusters in every situation we have tried. However, the clustering accuracies of cooperatively convergent algorithms are not as good as the purely penalized ones on this data set. Hence, the best Rand Index has been obtained by some other method in each case. When the initial k was set approaching to k^*, DSRPCL algorithm got the highest clustering accuracy. Nevertheless, the extra cluster had not been eliminated by it. Subsequently, when k was set at 10, the RPCCL obtained the highest RI value although its performance on cluster number estimate had degraded. As the value of k further increased, the performance of CoRe become much better. And when $k=20$, the CoRe obtained the best results, meanwhile, the CPCL's performance was in second place. It can be observed from the whole results that the performance of CPCL is much more robust to the settings of k.

Table 1
Clustering results on the Seeds data set ($k^*=3$).

k	Methods	#Clusters	PQ	RI	Time (#Epochs)
4	DSRPCL	4 ± 0.0	0.6623	0.8628	0.36 (94.85)
	RPCCL	**2.95** ±0.22	0.6849	0.8499	0.91 (100)
	CoRe	2.1 ± 0.31	0.5794	0.7593	1.04 (19.5)
	CCCL	2.85 ± 0.81	0.6273	0.8095	0.71 (77.35)
	CPCL	3.25 ± 0.55	**0.6922**	**0.8635**	0.56 (49.5)
10	DSRPCL	8.25 ± 1.12	0.3146	0.7673	1.61 (187.2)
	RPCCL	8.85 ± 1.18	0.3718	0.7763	9.06 (500)
	CoRe	2.45 ± 0.51	0.6385	0.8028	2.13 (28.75)
	CCCL	3.5 ± 0.82	0.6536	0.8442	3.68 (189.5)
	CPCL	**3.25** ±0.58	**0.7302**	**0.8840**	2.55 (110.9)
20	DSRPCL	17.05 ± 1.57	0.2020	0.7311	4.87 (296.15)
	RPCCL	18.3 ± 1.03	0.1783	0.7200	33.72 (1000)
	CoRe	2.7 ± 0.47	0.6738	0.8342	3.69 (39.7)
	CCCL	3.7 ± 0.92	0.6437	0.8329	13.71 (368.5)
	CPCL	**3.1** ±0.45	**0.7332**	**0.8771**	7.33 (168.3)

Table 2
Clustering results on the Wine data set ($k^*=3$).

k	Methods	#Clusters	PQ	RI	Time (#Epochs)
4	DSRPCL	4 ± 0.0	**0.7502**	**0.8977**	0.46 (125.35)
	RPCCL	2.8 ± 0.52	0.6686	0.8147	0.91 (100)
	CoRe	2.3 ± 0.65	0.5023	0.7082	1.52 (26.5)
	CCCL	3.45 ± 0.60	0.5520	0.7567	0.44 (55.15)
	CPCL	**3.15** ±0.58	0.6917	0.8332	0.65 (57.85)
10	DSRPCL	8.05 ± 0.99	0.4029	0.7941	1.73 (235.5)
	RPCCL	7.8 ± 1.51	0.5674	**0.8357**	9.41 (500)
	CoRe	2.6 ± 0.55	**0.6431**	0.8156	3.14 (39.2)
	CCCL	3.85 ± 1.18	0.5224	0.7602	1.61 (94.35)
	CPCL	**3.25** ±0.63	0.6382	0.8049	2.15 (93.26)
20	DSRPCL	20 ± 0.0	0.2091	0.7276	5.36 (324.1)
	RPCCL	18.65 ± 0.93	0.2070	0.7238	35.15 (1000)
	CoRe	**2.95** ±0.51	**0.6938**	**0.8503**	3.59 (39.5)
	CCCL	4.25 ± 1.21	0.4927	0.7535	6.80 (176.55)
	CPCL	3.25 ± 0.88	0.6316	0.8196	6.43 (145.65)

Table 3
Clustering results on the WDBC data set ($k^*=2$).

k	Methods	#Clusters	PQ	RI	Time (#Epochs)
3	DSRPCL	3 ± 0.0	0.6248	0.7553	0.47 (55.5)
	RPCCL	1.85 ± 0.36	0.4781	0.5553	2.11 (100)
	CoRe	2.15 ± 0.93	0.2664	0.5964	8.03 (26.2)
	CCCL	2.15 ± 0.36	0.7573	0.8321	0.72 (23.5)
	CPCL	**2** $\pm$ **0.0**	**0.7725**	**0.8415**	0.69 (20.4)
10	DSRPCL	9.7 ± 0.47	0.2111	0.5774	5.46 (225.8)
	RPCCL	5.9 ± 2.05	0.5136	0.6984	26.29 (500)
	CoRe	2.6 ± 1.31	0.2931	0.5719	23.51 (61.20)
	CCCL	1.95 ± 0.22	0.7215	0.8177	4.71 (47.15)
	CPCL	**2** $\pm$ **0.0**	**0.7551**	**0.8298**	2.63 (39.35)
20	DSRPCL	19.95 ± 0.22	0.1228	0.5311	20.99 (457.3)
	RPCCL	15.25 ± 1.86	0.1925	0.5629	96.67 (1000)
	CoRe	3.1 ± 0.91	0.3290	0.6126	49.51 (107.15)
	CCCL	1.95 ± 0.22	0.7267	0.8211	15.62 (82.05)
	CPCL	**2.05** ±0.22	**0.7582**	**0.8306**	7.97 (63.7)

As for the WDBC dataset, it contains 569 instances, whose 30 features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. 357 instances of them have the diagnosis of benign while the other 212 samples are regarded as malignant. From the summarized results in Table 3, we can find that the performance of CPCL algorithm is superior to all the other algorithms on this data set. Compared to the CCCL, which also had satisfying performance in this experiment, the CPCL converged more quickly and obtained better average partition quality with lower standard deviation. Moreover, since the sample size of WDBC data is much larger than the previous two data sets, the average computation load in each learning epoch of CoRe algorithm had an obvious increase, which implies that the running time of CoRe algorithm tends to be more sensitive to the sample size than that of the other methods.

4.1.2. Experiments on high dimensional data

In this sub-section, we further investigate the performance of CPCL algorithm on other data sets with higher dimensionality in comparison with the four counterparts. The utilized data sets are Sonar,[1] Control Chart[1], Multiple Features[1], and MNIST.[2] Table 4 has also given the basic information of them. For simplicity, the initial

[2] http://www.cs.nyu.edu/roweis/data.html

Table 4
Basic information of utilized data sets with higher dimensionality.

Data set	N	d	k^*
Sonar	208	60	2
Control chart	600	60	6
Multiple features	2000	649	10
MNIST	5000	784	10

Table 5
Clustering results on the Sonar data set ($k^*=2$).

k	Methods	#Clusters	PQ	RI	Time (#Epochs)
5	DSRPCL	4.7 ± 0.67	0.1591	0.5040	0.68 (130.4)
	RPCCL	3.5 ± 0.85	0.1934	0.5009	2.27(200)
	CoRe	1.65 ± 1.26	0.0331	0.5004	6.84 (32.6)
	CCCL	2.5 ± 0.71	0.2306	0.5032	0.92 (48.45)
	CPCL	**2.2** ±0.42	**0.2534**	**0.5058**	0.84 (42.15)

Table 6
Clustering results on the Control Chart data set ($k^*=6$).

k	Methods	#Clusters	PQ	RI	Time (#Epochs)
15	DSRPCL	12.4 ± 1.14	0.3288	0.8503	19.01 (386.55)
	RPCCL	7.85 ± 0.79	0.4161	0.8359	39.84 (500)
	CoRe	9.45 ± 1.43	0.3671	**0.8727**	80.57 (40.1)
	CCCL	2.5 ± 0.51	0.3741	0.7559	11.25 (93.15)
	CPCL	**5.35** ±0.61	**0.4603**	0.8618	12.58 (98.2)

Table 7
Clustering results on the Multiple Features data set ($k^*=10$).

k	Methods	#Clusters	PQ	RI	Time (#Epochs)
20	DSRPCL	20 ± 0.0	0.2021	**0.8765**	13.29 (56.25)
	RPCCL	14.75 ± 0.21	0.2116	0.8748	372.71 (500)
	CoRe	6.25 ± 1.56	0.1528	0.7864	66545.3 (40.6)
	CCCL	7.0 ± 0.54	0.2251	0.8476	197.69 (208.55)
	CPCL	**8.2** ±0.71	**0.2577**	0.8624	203.12 (201.5)

Table 8
Clustering results on the MNIST data set ($k^*=10$).

k	Methods	#Clusters	PQ	RI	Time (#Epochs)
20	DSRPCL	20 ± 0.0	0.1834	**0.8969**	246.87 (109.25)
	RPCCL	12.85 ± 1.42	0.2175	0.8245	1004.16 (500)
	CoRe	7.25 ± 1.83	0.1694	0.7835	267285.36 (52.65)
	CCCL	4.7 ± 2.36	0.1087	0.7552	362.71 (158.25)
	CPCL	**8.75** ±1.03	**0.2408**	0.8258	416.79 (176.8)

Table 9
Main statistics of utilized data sets.

Data set	N	d	k^*	σ
Sonar	208	60	2	2
USPS49	1673	256	2	10
Pendigit	3498	16	10	100
Multiple features	2000	649	10	2500

number of clusters k is set at not less than twice the true one in this experiment. The average clustering results of different algorithms from 20 trials on each data set have been summarized in Tables 5–8.

Table 10
Clustering results obtained by K-CPCL and kernel k-means methods.

Data set	k	Methods	#Clusters	PQ	RI
Sonar	2	Kernel k-means	2.0 ± 0.0	0.2627	0.5032
		K-CPCL	2.0 ± 0.0	0.2668	0.5006
	5	Kernel k-means	5.0 ± 0.0	0.1233	0.5034
		K-CPCL	2.5 ± 0.43	0.2315	0.5047
USPS49	2	Kernel k-means	2.0 ± 0.0	0.4867	0.6561
		K-CPCL	2.0 ± 0.0	0.4779	0.6485
	5	Kernel k-means	5.0 ± 0.0	0.2223	0.5795
		K-CPCL	2.35 ± 0.63	0.4463	0.6329
Pendigit	10	Kernel k-means	9.8 ± 0.52	0.4504	0.9122
		K-CPCL	9.65 ± 0.42	0.4438	0.9015
	20	Kernel k-means	18.3 ± 0.80	0.4344	0.9347
		K-CPCL	11.7 ± 1.06	0.4556	0.9070
Multiple features	10	Kernel k-means	7.2 ± 1.15	0.2805	0.8621
		K-CPCL	9.5 ± 0.44	0.3121	0.8875
	20	Kernel k-means	13.8 ± 1.34	0.2863	0.8882
		K-CPCL	10.7 ± 0.85	0.3316	0.8965

It can be observed from these tables that the proposed CPCL algorithm has given the most approximate estimate for the number of clusters on each utilized data set. Although in some cases, the Rand Index value obtained by CPCL is not the best one, its Partition Quality result is superior to the other counterparts. This indicates that, for the data sets with relatively higher dimensionality, the CPCL algorithm also has a good ability in clustering analysis without knowing cluster number as it can simultaneously learn the cluster number and cluster membership well. Moreover, for the Multiple Features and MNIST data sets with both of large sample size and high dimensionality, the running time of CPCL algorithm is still acceptable. By contrast, the running time of CoRe algorithm in these two cases is hundreds of times more than that of others, which is too expensive for practical application.

4.2. Performance of K-CPCL algorithm

To investigate the performance of K-CPCL algorithm, we applied it to four benchmark data sets, i.e., Sonar[1], USPS49,[3] Pendigit[1], and Multiple Features[1], and compared its results to that obtained by standard kernel k-means method [16]. The general information of the four data sets along with the chosen value of σ in the Gaussian kernel function for each data set has been given in Table 9. In the experiments, each algorithm has executed 20 times under different settings of k. The learning rate η in K-CPCL algorithm was set at 0.0001. Partition Quality (PQ) and Rand Index (RI) were utilized to evaluate the clustering accuracy and the learnt number of clusters under each situation has also been recorded for comparison.

Table 10 shows the clustering results obtained in this experiment. From the table, we can find that when the cluster number was initialized equal to the true value, the performance of K-CPCL algorithm matched that of kernel k-means method. However, when the initial value of cluster number was set much larger than the true one, the performance of kernel k-means degraded significantly because it cannot recognize the extra clusters. In the experiment, the cluster number learnt by kernel k-means was sometimes less than the setting value is due to the generation of empty clusters. By contrast, the proposed K-CPCL algorithm can give a good estimate of the cluster number during the clustering

[3] http://www-stat.stanford.edu/tibs/ElemStatLearn/data.html

Table 11
Comparing the performance of CPCL and K-CPCL methods.

Data set	k	Methods	#Clusters	PQ	RI
Sonar	5	CPCL	2.2 ± 0.42	0.2534	0.5058
		K-CPCL	2.5 ± 0.43	0.2315	0.5047
*USPS*49	5	CPCL	1.25 ± 0.37	0.1072	0.5469
		K-CPCL	2.35 ± 0.63	0.4463	0.6329
Pendigit	20	CPCL	7.35 ± 1.36	0.3452	0.8537
		K-CPCL	11.7 ± 1.06	0.4556	0.9070
Multiple features	20	CPCL	8.2 ± 0.71	0.2577	0.8624
		K-CPCL	10.7 ± 0.85	0.3316	0.8965

process. Therefore, the superiority of K-CPCL method is more obvious in unsupervised clustering analysis without knowing true cluster number.

Moreover, to investigate the advantage of kernel method, we further compare the performance of CPCL and K-CPCL algorithms in Table 11. It can be observed that the cluster number and cluster membership learnt by the K-CPCL algorithm with kernel method are more accurate in most cases. Especially, for the USPS49 data set, the results of K-CPCL are much better than that of the CPCL method as CPCL algorithm usually converges to only one cluster on this data set. Therefore, the K-CPCL algorithm is expected to have more robust performance in unsupervised clustering analysis on the data sets with complex cluster structure.

5. Conclusion

In this paper, we have presented a novel competitive learning model named Cooperative and Penalized Competitive Learning (CPCL), which performs the competition with the two different kinds of mechanisms simultaneously: cooperation and penalization. On one hand, the cooperation mechanism enables the closest seed points to update together and gradually converge to the corresponding cluster centers, which gives the algorithm good convergence speed and high precision. On the other hand, the penalization mechanism provides the other seed points with the opportunity to wander in the clustering space, which enables it to perform the clustering problem with the robustness against the initialization of the seed points and the overlap of the data clusters. Furthermore, to solve the cluster number selection problem in nonlinear clustering, we have introduced the proposed competition mechanism into kernel clustering framework and presented a new kernel-based competitive learning algorithm. Numerical studies have shown the efficacy of the proposed approach.

Conflict of interest

None declared.

Acknowledgment

This work was supported by the Faculty Research Grant of Hong Kong Baptist University (HKBU) under Project FRG2/12-13/082, the National Science Foundation of China under Grant 61272366, and the Strategic Development Fund of HKBU: 03-17-033.

References

[1] B. Zhang, C. Zhang, X. Yi, Competitive EM algorithm for finite mixture models, Pattern Recognit. 37 (1) (2004) 131–144.
[2] H. Frigui, R. Krishnapuram, A robust competitive clustering algorithm with applications in computer vision, IEEE Trans. Pattern Anal. Mach. Intell. 21 (5) (1999) 450–465.
[3] A.W.-C. Liew, H. Yan, M. Yang, Pattern recognition techniques for the emerging field of bioinformatics: a review, Pattern Recognit. 38 (11) (2005) 2055–2073.
[4] J.B. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, 1967, pp. 281–297.
[5] G.J. McLachlan, T. Krishnan, The EM Algorithm and Extensions, Wiley, New York, 1997.
[6] G. Hamerly, C. Elkan, Learning the k in k-means, in: Proceedings of the 17th Annual Conference on Neural Information Processing Systems (NIPS), 2003, pp. 281–288.
[7] L. Xu, A. Krzyzak, E. Oja, Rival penalized competitive learning for clustering analysis, RBF net, and curve detection, IEEE Trans. Neural Netw. 4 (4) (1993) 636–648.
[8] Y.M. Cheung, On rival penalization controlled competitive learning for clustering with automatic cluster number selection, IEEE Trans. Knowl. Data Eng. 17 (11) (2005) 1583–1588.
[9] Y.M. Cheung, Maximum weighted likelihood via rival penalized EM for density mixture clustering with automatic model selection, IEEE Trans. Knowl. Data Eng. 17 (6) (2005) 750–761.
[10] J. Ma, T. Wang, A cost-function approach to rival penalized competitive learning (RPCL), IEEE Trans. Syst. Man Cybern.—Part B: Cybern. 36 (4) (2006) 722–737.
[11] D. Bacciu, A. Starita, Competitive Repetition Suppression (CoRe) clustering: a biologically inspired learning model with application to robust clustering, IEEE Trans. Neural Netw. 19 (11) (2008) 1922–1941.
[12] Y.M. Cheung, A competitive and cooperative learning approach to robust data clustering, in: Proceedings of IASTED International Conference on Neural Networks and Computational Intelligence, 2004, pp. 131–136.
[13] T. Li, W.J. Pei, S.P. Wang, Y.M. Cheung, Cooperation controlled competitive learning approach for data clustering, in: Proceedings of International Conference on Computational Intelligence and Security, 2008, pp. 24–29.
[14] J. Shawe-Taylor, N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, 2004.
[15] M. Filippone, F. Camastra, F. Masulli, S. Rovetta, A survey of kernel and spectral methods for clustering, Pattern Recognit. 41 (2008) 176–190.
[16] B. Schölkopf, A. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (5) (1998) 1299–1319.
[17] G.F. Tzortzis, A.C. Likas, The global kernel k-means algorithm for clustering in feature space, IEEE Trans. Neural Netw. 20 (7) (2009) 1181–1194.
[18] D. Macdonald, C. Fyfe, The kernel self-organising map, in: Proceedings of the Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies, vol. 1, 2000, pp. 317–320.
[19] R. Inokuchi, S. Miyamoto, LVQ clustering and SOM using a kernel function, in: Proceedings of IEEE International Conference on Fuzzy Systems, vol. 3, 2004, pp. 1497–1500.
[20] S.C. Ahalt, A.K. Krishnamurty, P. Chen, D.E. Melton, Competitive learning algorithms for vector quantization, Neural Netw. 3 (3) (1990) 277–291.
[21] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, B. Schölkopf, An introduction to kernel-based learning algorithms, IEEE Trans. Neural Netw. 12 (2) (2001) 181–201.

Hong Jia received the B.S. and master degrees from Huazhong University of Science and Technology, China, in 2008 and 2010, respectively. She got the Ph.D. degree in computer science from Hong Kong Baptist University in 2013. Her research interests are in the areas of pattern recognition, machine learning and data mining.

Yiu-Ming Cheung (SM'06) is a Professor at Department of Computer Science in Hong Kong Baptist University. He received the Ph.D. degree from the Department of Computer Science and Engineering, The Chinese University of Hong Kong in 2000, and then joined the Department of Computer Science at Hong Kong Baptist University in 2001. His current research interests are in the fields of machine learning and information security, particularly the topics on clustering analysis, blind source separation, neural networks, nonlinear optimization, watermarking and lip-reading. He is the founding Chairman of IEEE (Hong Kong) Computational Intelligence Chapter. Currently, he is also the Associate Editor of *IEEE Transactions on Neural Networks and Learning Systems*, *Knowledge and Information Systems*, as well as the guest co-editor and editorial board member of the several international journals.

Jiming Liu is Chair Professor in Computer Science and Associate Dean of Faculty of Science at Hong Kong Baptist University. He is a Fellow of the IEEE. Prof. Liu received his M.Eng. and Ph.D. degrees from McGill University in Montreal, Canada. His current research focuses on Data Mining and Data Analytics, Multi-Agent Computing, Collective Intelligence, Health Informatics, and Computational Epidemiology. Prof. Liu has served as the Editor-in-Chief of Brain Informatics: Brain Data Computing and Health Studies (Springer) and Web Intelligence and Agent Systems (IOS), and an Associate Editor of IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on Cybernetics, Big Data and Information Analytics (AIMS), Computational Intelligence (Wiley), and Neuroscience and Biomedical Engineering (Bentham), among others.