A Cost-Function Approach to Rival Penalized Competitive Learning (RPCL)

Jinwen Ma and Taijun Wang

***Abstract*—Rival penalized competitive learning (RPCL) has been shown to be a useful tool for clustering on a set of sample data in which the number of clusters is unknown. However, the RPCL algorithm was proposed heuristically and is still in lack of a mathematical theory to describe its convergence behavior. In order to solve the convergence problem, we investigate it via a cost-function approach. By theoretical analysis, we prove that a general form of RPCL, called distance-sensitive RPCL (DSRPCL), is associated with the minimization of a cost function on the weight vectors of a competitive learning network. As a DSRPCL process decreases the cost to a local minimum, a number of weight vectors eventually fall into a hypersphere surrounding the sample data, while the other weight vectors diverge to infinity. Moreover, it is shown by the theoretical analysis and simulation experiments that if the cost reduces into the global minimum, a correct number of weight vectors is automatically selected and located around the centers of the actual clusters, respectively. Finally, we apply the DSRPCL algorithms to unsupervised color image segmentation and classification of the wine data.**

***Index Terms*—Clustering analysis, competitive learning (CL), convergence, cost function, gradient descent.**

I. Introduction

As an adaptive version of the classical k-means clustering, competitive learning (CL) has been developed for unsupervised learning in artificial neural networks and provided us a promising tool for clustering, pattern recognition, and vector quantization [1]–[3]. It is also useful to improve the training of multilayer feedforward networks [4]–[6]. The original CL algorithm (or rule), now known as the classical (or simple) CL algorithm, was proposed and studied from the early 1960s [7]. In the late 1980s, it was found by Rumelhart and Zipser [8], Grossberg [9], and Hecht-Nielsen [10] that the algorithm has the so-called dead-unit (or underutilized) problem. Many efforts have been made to solve this problem. A typical strategy is reducing the learning rate of the frequent winners [9], [11]–[13], sometimes called conscience mechanism [12]. The frequency-sensitive CL (FSCL) algorithm [13] is an example that does improve the classical CL considerably [15], [44].

Manuscript received June 17, 2004; revised February 1, 2005, July 29, 2005, and November 3, 2005. This work was supported by the Natural Science Foundation of China under Projects 60471054 and 60071004. This paper was recommended by Associate Editor N. Pal.

J. Ma is with the Department of Information Science, School of Mathematical Sciences and the Laboratory of Mathematics and Applied Mathematics, Peking University, Beijing 100871, China (e-mail: jwma@math.pku.edu.cn).

T. Wang is with the Department of Radio Engineering, Southeast University, Nanjing 210018, China.

Digital Object Identifier 10.1109/TSMCB.2006.870633

It has been further addressed by Xu *et al.* in [14] and [15] that all existing CL algorithms have another critical problem: The selection of an appropriate number of units. As well known with the k-means algorithm, the number k of clusters should be appropriately preselected; otherwise, the algorithm will perform badly. In a CL-related network, k directly corresponds to an externally preselected number for the units used in a neural network, e.g., the number either corresponds to the number of resulted clusters when the network is used for clustering analysis or to the number of radial basis functions (RBFs) when CL is used in an RBF network. The experiments in [14] and [15] have shown that an inappropriately selected k will result in a poor clustering that will in turn affect the performances of FSCLs.

To tackle this problem, the rival-penalized-CL (RPCL) algorithm was proposed in [14] and [15] by adding a new mechanism into an FSCL. For each input sample, the basic idea is that not only the weight vector of the winner unit is modified to adapt to the input, but also the weight vector of its rival (the second winner) is delearned by a smaller learning rate. Many experimental results have shown that RPCL automatically allocates an appropriate number of units for an input data set when they are used for clustering. Later, the RPCL algorithm has been applied to clustering, vector quantization, training of RBF neural networks, plant diagnosis, and financial prediction (e.g., [16]–[27]). Methodologically, the RPCL algorithm was used to establish some new mechanisms for information processing [28]–[30]. Furthermore, in [31]–[34], the RPCL algorithm has also been generalized to several versions for different types of sample data.

The RPCL algorithm was proposed heuristically. It has been shown that RPCL can be regarded as a fast approximate implementation of a special-case Bayesian Ying Yang (BYY) harmony learning on a Gaussian mixture; thus, the ability of selecting a number of clusters can be understood from the model-selection ability of BYY harmony learning [35], [36]. However, there still lacks a mathematical theory to directly describe the convergence and, particularly, the correct convergence behavior of RPCL, and those existing convergence theories for the classical CL and FSCL algorithms under certain constraints [37]–[41] are hardly possible to be adopted since RPCL demonstrates a very different convergence behavior. The key issue is whether the RPCL algorithm can automatically select a correct number of the weight vectors that converge to each center of the clusters, respectively, while driving all the other extra weight vectors far away from the sample data. This correct convergence problem is difficult, and there has not been any mathematical theory to solve such kind of convergence problem

yet, to the best of our knowledge. Recently, based on the maximum weighted likelihood learning framework, Cheung [42] proposed the rival-penalized-EM (RPEM) algorithm that can be considered as a generalized version of the RPCL algorithm, with the feature of automatically selecting an appropriate number of densities in density mixture clustering. Although the RPEM algorithm can outperform the RPCL algorithm in ceratin aspects, there is also a lack of a mathematical theory to describe why the RPEM or the RPCL algorithm can converge correctly for determining the appropriate number of clusters in the sample data.

In this paper, we investigate the problem of RPCL correct convergence via a cost-function approach. By analyzing the characteristics of the RPCL algorithm, we establish a cost function by which we can derive a general form of the RPCL algorithm, called distance-sensitive RPCL (DSRPCL) algorithm. That is, DSRPCL is associated with the minimization of the cost function on the weight vectors in a CL network. We prove that when the cost function reduces to a local minimum, a number of weight vectors fall into a hypersphere surrounding the sample data, while the other weight vectors diverge to infinity. Moreover, it is further shown by the theoretical analysis and simulation experiments that if this minimum is global, the DSRPCL process consists of first making a correct number of weight vectors fall into the hypersphere and then further pushing them to approach each center of the clusters, respectively, with these extra weight vectors driven to infinity. Furthermore, we test the DSRPCL algorithm on the unsupervised color image segmentation and the classification of the wine dataset.

The problem of RPCL correct convergence is addressed in Section II. Then, a cost function is constructed for a general form of the RPCL algorithm, i.e., the DSRPCL algorithm, in Section III. In Section IV, we further analyze the convergence properties of the DSRPCL processes in using the cost function. In Section V, simulation and application experiments are demonstrated, and the conclusion is presented in Section VI.

II. Problem of RPCL Correct Convergence

We begin with the classical CL algorithm. Given a CL network, i.e., a layer of units with the output of each unit denoted by u_i and its weight vector by $W_i = [w_{i1}, \ldots, w_{id}]^{\mathrm{T}}$ for $i = 1, \ldots, n$, the classical CL algorithm consists of the following two steps.

Step 1) Randomly take a sample X from a sample data set $\mathcal{S} \subset R^d$; for $i = 1, \ldots, n$, let

$$u_i = \begin{cases} 1, & \text{if } i = c \text{ such that } \|X - W_c\| = \min_j \|X - W_j\| \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where $\|\cdot\|$ denotes the Euclidean norm.

Step 2) Update the weight vectors W_i by

$$\Delta W_i = \alpha_i u_i (X - W_i) \tag{2}$$

which is often called the winner-takes-all rule since only the winning unit W_c is updated. The parameter α_c with $0 \le \alpha_c \le 1$ is the learning rate that either is a small positive number or starts from a reasonable initial value and then reduces to zero according to the so-called Robbin–Monro stochastic approximation procedure [43]

$$\lim_{t\to\infty} \alpha_c(t) = 0 \qquad \sum_{t=1}^{\infty} \alpha_c(t) = \infty. \tag{3}$$

A typical example of the learning rate satisfying (3) is $\eta(t) = \eta_0/(c_1 t + c_0)$, where η_0, c_0, and c_1 are some positive constants.

The FSCL algorithm is a straightforward extension of the classical CL algorithm, obtained by modifying (1) into the following.

$$u_i = \begin{cases} 1, & \text{if } i = c \text{ such that} \\ & \quad \gamma_c \|X - W_c\| = \min_j (\gamma_j \|X - W_j\|) \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where γ_j is the winning frequency of unit j, i.e., $\gamma_j = t_j / \sum_{i=1}^{k} t_i$ and t_i is the cumulative number of the occurrences of $u_i = 1$.

Furthermore, we get the RPCL algorithm by adding the rival penalizing mechanism to FSCL as following.

Step 1) Randomly take a sample X from $\mathcal{S}$; for $i = 1, \ldots, n$, let

$$u_i = \begin{cases} 1, & \text{if } i = c \text{ such that} \\ & \quad \gamma_c \|X - W_c\| = \min_j (\gamma_j \|X - W_j\|) \\ -1, & \text{if } i = r \text{ such that} \\ & \quad \gamma_r \|X - W_r\| = \min_{j \ne c} (\gamma_j \|X - W_j\|) \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Step 2) Update the weight vectors W_i by

$$\Delta W_i = \begin{cases} \alpha_c (X - W_i), & \text{if } u_i = 1 \\ -\alpha_r (X - W_i), & \text{if } u_i = -1 \\ 0, & \text{otherwise} \end{cases}$$

where $0 \le \alpha_c$ and $\alpha_r \le 1$ are the learning and delearning rates for the winner unit c and the rival unit r, respectively. It was assumed by Xu *et al.* in [14] and [15] that α_c and α_r reduce with t to zero with $\alpha_r \ll \alpha_c$ at each step. Moreover, from the probabilistic perspective, several other variants of the RPCL algorithm have been proposed for different types of sample data [32]–[38]. Since the fundamental mechanism behind all the studies remains the same, in this paper, we only focus on the RPCL algorithm in its original version (5).

As the learning and delearning rates α_c and α_r reduce to zero at a suitable rate, e.g., in the way of (3), the RPCL algorithm will either converge to or freeze in a limit—a group of weight vectors. In this sense, the convergence of the RPCL algorithm is trivial. Instead, the key problem is whether it converges correctly or not. For a set of sample data with k clusters and $n \ge k$, the problem of RPCL correct convergence is whether it can automatically make k weight vectors converge to each center of the k clusters, respectively, while driving the other extra weight vectors far away from the sample data.

To clearly describe how a weight vector is driven away from the sample data, we have the aid of a (bounded) hypersphere $\mathcal{G}$ that can contain all the sample data within it. Provided that the RPCL algorithm starts with a set of initial values of $W_1^{(0)}, \ldots, W_n^{(0)}$ within $\mathcal{G}$, the correct convergence of the RPCL algorithm can be described through the following three properties.

1) **Separation Nature**. After sufficient iterations of the RPCL algorithm, each weight vector will finally either fall into $\mathcal{G}$ or remain to stay outside $\mathcal{G}$ and never get into $\mathcal{G}$. That is, after a number of iterations, a number of weight vectors will be always in $\mathcal{G}$ and the other weight vectors will be always outside $\mathcal{G}$. Thus, the hypersphere separates the weight vectors into two groups.
2) **Correct Division**. The number of the weight vectors fallen into $\mathcal{G}$ should be k, i.e., the number of the actual clusters in the sample data. Obviously, only in this case, it is possible for the RPCL process to result in a reasonable classification.
3) **Correct Location**. k weight vectors will finally converge to or locate around each center of the k actual clusters, respectively.

Obviously, property 1) is a prerequisite to the RPCL correct convergence. When property 1) holds, it is possible to check whether a weight vector will finally be driven away from the data set or not. When properties 1) and 2) both hold, we are able to determine the number of clusters via the number of weight vectors fallen within the hypersphere $\mathcal{G}$.

In the following, we establish a cost function based on which we can derive a general form of the RPCL algorithm and investigate the correct convergence problem from the perspective of the cost minimization.

III. Cost Function for the RPCL Algorithm

In this section, according to the characteristics of the RPCL algorithm we introduce a new kind of cost function and then consider its derivatives. Moreover, the gradient-descent algorithm of the cost function is constructed and analyzed, being shown to be a general form of the RPCL algorithm.

A. Idea and Definition of the Cost Function

Given a set of sample data $\mathcal{S} = \{X^\mu\}_{\mu=1}^N$ with $X^\mu = [x_1^\mu, x_2^\mu, \ldots, x_d^\mu]^{\mathrm{T}}$, the classical CL or FSCL algorithm can be regarded as the adaptive versions of the well-known k-mean algorithm, which minimizes the mean-square-error (MSE) cost function as follows:

$$\begin{aligned} E_{\mathrm{MSE}}(\mathbf{W}) &= \frac{1}{2}\sum_{ij\mu} M_i^\mu \left(x_j^\mu - w_{ij}\right)^2 \\ &= \frac{1}{2}\sum_{i\mu} M_i^\mu \|X^\mu - W_i\|^2 \\ &= \frac{1}{2}\sum_{\mu} \left\|X^\mu - W_{c(\mu)}\right\|^2 \end{aligned} \tag{6}$$

where $\mathbf{W} = [W_1, W_2, \ldots, W_k]$, i.e., $n = k$, and each $W_i = [w_{i1}, w_{i2}, \ldots, w_{id}]^{\mathrm{T}}$ is the weight vector of the unit i. Moreover, we have

$$M_i^\mu = \begin{cases} 1, & \text{if } i = c(\mu) \text{ such that} \\ & \quad \left\|X^\mu - W_{c(\mu)}\right\| = \min_j \|X^\mu - W_j\| \\ 0, & \text{otherwise} \end{cases}$$

with $c(\mu)$ denoting the index of the winner unit (or its weight vector) for the μth sample. If there are two or more weight vectors that are equidistant and closest to the sample X^μ, $c(\mu)$ denotes the least of the indexes of these weight vectors. Therefore, M_i^μ is a cluster membership function for each input sample X^μ. That is, X^μ belongs to the ith cluster if it is one; and X^μ does not belong to the ith cluster if it is zero.

However, $E_{\mathrm{MSE}}(\mathbf{W})$ cannot be used for deciding a correct number k, because it decreases monotonically as the number of the units increases. Thus, it does not apply to the RPCL algorithm. In order to construct a suitable cost function to describe the RPCL algorithm, we consider its mechanism from two perspectives. First, those areas where samples locate densely will strongly attract the weight vectors. Second, each weight vector will push the other weight vectors away from itself, which makes it possible to drive the extra weight vectors far away from the sample data. Moreover, when a weight vector diverges to infinity, the corresponding cluster becomes empty and can be neglected. In view of these considerations, we are motivated to consider the following cost function:

$$E(\mathbf{W}) = E_1(\mathbf{W}) + E_2(\mathbf{W}) \tag{7}$$

where

$$E_1(\mathbf{W}) = E_{\mathrm{MSE}}(\mathbf{W}) = \frac{1}{2}\sum_{\mu} \left\|X^\mu - W_{c(\mu)}\right\|^2$$

$$E_2(\mathbf{W}) = \frac{2}{P}\sum_{\mu, i \neq c(\mu)} \|X^\mu - W_i\|^{-P}$$

where $\mathbf{W} = \mathrm{vec}[W_1, W_2, \ldots, W_n]$ and P is a positive number. This new cost function is not only proportional to the distance between each sample point and its winner weight vector where $c(\mu) = 1$ but also inversely proportional to the distance between this sample point and any other loser weight vector where $c(\mu) = 0$. It will be shown by both analysis and experiment in the following sections that a lower value of $E(\mathbf{W})$ corresponds to a better clustering solution $\mathbf{W}$. Moreover, the global minimum value of $E(\mathbf{W})$ decreases with n from the beginning, i.e., $n = 1$, and then does not decrease with n beyond the correct number k. From this cost function, we can derive a general form of the RPCL algorithm via the gradient-descent method.

B. Derivatives of the Cost Function

According to the definition given by (7), the cost function becomes positive infinity when two weight vectors are equal and meet at a sample X^μ. We let $\mathcal{B}$ denote the set of these special $\mathbf{W}$. Clearly, the cost function is not differentiable in $\mathcal{B}$. However, except $\mathcal{B}$, it can be easily found from (7) that the

cost function is finite and continuous at any $\mathbf{W}$. Moreover, it can be proved that the cost function is differentiable at any $\mathbf{W}$ excluding $\mathcal{B}$.

In order to do so, we begin to consider the differentiablity of the cost function in the general case of $\mathbf{W}$ where there is just a unique weight vector closest to a sample X^μ. In the other words, each sample is closest to a unique weight vector. As the samples are fixed, the membership functions M_i^μ can remain unchanged in a neighborhood of $\mathbf{W}$. In this case, $E(\mathbf{W})$ is smooth within the neighborhood and thus differentiable at $\mathbf{W}$. Moreover, the derivative of the cost function with respect to w_{ij} is given by

$$\begin{aligned}\frac{\partial E(\mathbf{W})}{\partial w_{ij}} &= \frac{\partial E_1(\mathbf{W})}{\partial w_{ij}} + \frac{\partial E_2(\mathbf{W})}{\partial w_{ij}} \\ &= -\sum_{\mu} \delta_{i,c(\mu)} \left(x_j^\mu - w_{ij}\right) + \sum_{\mu,i} \left(1 - \delta_{i,c(\mu)}\right) \\ &\quad \times \|X^\mu - W_i\|^{-P-2} \left(x_j^\mu - w_{ij}\right) \end{aligned} \tag{8}$$

where $\delta_{i,j}$ is the Kronecker number.

We further consider the differentiablity of the cost function in the special case of $\mathbf{W}$ where two or more weight vectors are equidistant and closest to a sample X^μ. We can consider this kind of $\mathbf{W}$ as a point at the boundaries of the regions (in the whole weight definition space R^{nd}) where all the membership functions M_i^μ do not change. Suppose that $\mathbf{W}'$ is just such a boundary point where there are only two component weight vectors W_i' and W_j' (with $i < j$) that are equidistant and closest to a sample $X^{\mu'}$. That is

$$\left\|W_i' - X^{\mu'}\right\| = \left\|W_j' - X^{\mu'}\right\| = \min_l \left\|W_l' - X^{\mu'}\right\| > 0.$$

In this case, a neighborhood of $\mathbf{W}'$ is divided by the smooth (curved) surface $\|W_i - X^{\mu'}\| = \|W_j - X^{\mu'}\|$ into two parts or subregions $\mathcal{A}_{l_i}$ and $\mathcal{A}_{l_j}$, respectively, with the constraints $\|W_i - X^{\mu'}\| = \min_l \|W_l - X^{\mu'}\| \le \|W_j - X^{\mu'}\|$ in $\mathcal{A}_{l_i}$, and $\|W_j - X^{\mu'}\| = \min_l \|W_l - X^{\mu'}\| < \|W_i - X^{\mu'}\|$ in $\mathcal{A}_{l_j}$. Since $i < j$, according to the definition of the winning unit, the above dividing surface is within $\mathcal{A}_{l_i}$.

From the side of $\mathcal{A}_{l_i}$, the cost function is differentiable at $\mathbf{W}'$ and has the derivatives given by (8) with $M_i^{\mu'} = 1$ and $M_j^{\mu'} = 0$. From the other side of $\mathcal{A}_{l_j}$, the cost function is also differentiable at $\mathbf{W}'$ and has the derivatives given by (8) with $M_i^{\mu'} = 0$ and $M_j^{\mu'} = 1$. Because $\|W_i' - X^{\mu'}\| = \|W_j' - X^{\mu'}\|$, it can be easily verified that the left-hand and right-hand derivatives are equal. Because each direction toward $\mathbf{W}'$ is either within $\mathcal{A}_{l_i}$ or within $\mathcal{A}_{l_j}$, the derivatives of the cost function at $\mathbf{W}'$ lead to the same values in all the directions. According to the differential theory, the cost function is differentiable at this boundary point $\mathbf{W}'$ and its derivative takes the form of (8).

In a similar way, we can prove that the cost function is differentiable at those boundary points where there are three or more weight vectors that are equidistant and closest to a sample or there are two or more samples in this equidistant situation. Therefore, the cost function is differentiable at all the boundary points except $\mathcal{B}$. Summing up the above results, we obtain that the cost function is differentiable at any $\mathbf{W}$ excluding $\mathcal{B}$.

C. DSRPCL Algorithm

Based on the derivatives of the cost function, we can construct a gradient-descent algorithm as follows:

$$\Delta w_{ij} = -\eta \frac{\partial E(\mathbf{W})}{\partial w_{ij}} \tag{9}$$

where η is a small positive number as the learning rate. For convenience of analysis, we define the gradient of $E(\mathbf{W})$ by

$$\operatorname{grad}\left(E(\mathbf{W})\right) = \left[\frac{\partial E(\mathbf{W})}{\partial w_{11}}, \frac{\partial E(\mathbf{W})}{\partial w_{12}}, \ldots, \frac{\partial E(\mathbf{W})}{\partial w_{nd}}\right]^{\mathrm{T}}.$$

In this case, the gradient-descent algorithm can be also expressed by

$$\Delta \mathbf{W} = -\eta \operatorname{grad}\left(E(\mathbf{W})\right). \tag{10}$$

We let the algorithm start from a set of initial weight vectors $\mathbf{W}^{(0)} = \operatorname{vec}[W_1^{(0)}, \ldots, W_n^{(0)}]$ in the field $R^{nd} - \mathcal{B}$. As η is small enough, $E(\mathbf{W})$ will decrease after each modification. Thus, $E(\mathbf{W})$ cannot become infinity. That is, the updated weight vectors $\mathbf{W}^{(t)} = \mathbf{W}^{(t-1)} - \eta \operatorname{grad}(\mathbf{W}^{(t-1)})$ will never enter $\mathcal{B}$. On the other hand, we can stop the algorithm when the change of $E(\mathbf{W})$ between two consecutive steps is less than a threshold value.

Instead of this batch algorithm, we can get the following adaptive algorithm from (8), i.e., for the current sample X^μ, we update

$$\Delta W_i = \begin{cases} \eta(X^\mu - W_i), & \text{if } i = c(\mu) \\ -\eta \|X^\mu - W_i\|^{-P-2}(X^\mu - W_i), & \text{otherwise} \end{cases} \tag{11}$$

which is actually a CL algorithm. At each sample X^μ, the weight vector $W_{c(\mu)}$ of the winner unit $c(\mu)$ is modified by

$$\Delta W_{c(\mu)} = \eta(X^\mu - W_i)$$

which can be regarded as winner learning. On the other hand, all the loser weight vectors are modified by

$$\Delta W_i = -\eta \|X^\mu - W_i\|^{-P-2}(X^\mu - W_i), \qquad \text{for } i \neq c(\mu)$$

which can be regarded as a delearning action.

For convenience, we can divide the sample space R^d into n regions as follows:

$$R_i = \left\{X \in R^d : \|X - W_i\| \le \|X - W_j\|, j \neq i\right\}, \qquad \text{for } i = 1, 2, \ldots, n. \tag{12}$$

Accordingly, the sample data set $\mathcal{S}$ is divided into n clusters (or classes) by

$$C_i = \mathcal{S} \cap R_i, \qquad \text{for } i = 1, \ldots, n. \tag{13}$$

Particularly, if a sample point X^μ exactly lies on a boundary of some regions, we break the ties by classifying it as any of the corresponding clusters. Furthermore, we define $\overline{C}_i = \mathcal{S} - C_i$ as the complementary set of the cluster C_i.

Comparing the derived algorithm with the classical CL algorithm (1), we can observe that the mechanism of the algorithm (11) is that the modification of the winner weight vectors reduces $E_1(\mathbf{W})$, which drives a group of weight vectors to converge to the centers of the actual clusters in the sample data, while the modification of the loser (or other) weight vectors reduces $E_2(\mathbf{W})$, which drives the remaining weight vectors away from the winner weight vectors so that the extra weight vectors are pushed away from the sample data. Therefore, this algorithm behaves in a similar way as the RPCL algorithm.

We further explore the relation between this algorithm and the original RPCL algorithm. Observing the sum of $E_2(\mathbf{W})$ over the sample X^μ, it can be found that the dominant term is just $\|X^\mu - W_{r(\mu)}\|^{-P}$, where $r(\mu)$ denotes the index of the rival unit. We can modify this rival weight vector such that $E_2(\mathbf{W})$ is affected by the largest term only. The algorithm has the following form:

$$\Delta W_i = \begin{cases} \eta(X^\mu - W_i), & \text{if } i = c(\mu) \\ -\eta\|X^\mu - W_i\|^{-P-2}(X^\mu - W_i), & \text{if } i = r(\mu) \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

which takes the same form as the RPCL algorithm by letting

$$\alpha_c = \eta, \qquad \alpha_r = \eta \left\|X^\mu - W_{r(\mu)}\right\|^{-P-2}.$$

Furthermore, we have

$$\frac{\alpha_c}{\alpha_r} = \left\|X^\mu - W_{r(\mu)}\right\|^{2+P}. \tag{15}$$

That is, α_c/α_r is strongly dependent on $\|X^\mu - W_{r(\mu)}\|$. Moreover, given the parameter P, the problem of choosing the learning rate α_c and the delearning rate α_r is equivalent to choosing the learning rate η.

In summary, a cost-function formulation of the general form of the RPCL algorithm has been established. Since this form of the algorithm is distance sensitive, we refer to it as the DSRPCL algorithm. For clarity, the algorithm given by (9) or (10) is called the batch DSRPCL algorithm. The adaptive algorithm given by (11) is called the DSRPCL1 algorithm, which implements the delearning action to all the losers. The adaptive algorithm given by (14) is called the DSRPCL2 algorithm, which implements the delearning action only to the rival. As a kind of gradient learning algorithm, the learning rate η plays an important role in the DSRPCL algorithm. In fact, it is just the convergence rate of this linearly convergent learning algorithm. Specifically, when η is small, the DSRPCL algorithm converges slowly but stably. Otherwise, when η is large, the DSRPCL algorithm may converge quickly but often be trapped into a local minimum. In the batch DSRPCL algorithm, η can be a small positive constant or searched at each step in the optimum manner. However, in the DSRPCL 1 and 2 algorithms, according to the stochastic approximation procedure [43], η should start from a reasonable initial value and then reduce to zero in the way of (3). Practically, the learning rate can be selected as a positive constant by experience. As will be shown by the simulation experiments in Section V, the DSRPCL algorithm is effective and has the same convergence behavior as the original RPCL algorithm. Therefore, in the rest of this paper, we study the convergence properties of the RPCL learning based on the DSRPCL algorithm.

Before doing so, there are still two points to be further explained. First, the winning frequencies of the units do not appear in the DSRPCL algorithm, that is, the conscience mechanism is not used. Actually, the conscience mechanism is a constraint that all the winning frequencies are equal. Its purpose is to solve the dead unit problem, which, in this paper, is alternatively avoided by selecting the initial weight vectors randomly within the hypersphere $\mathcal{G}$ or as a group of sample points. Moreover, for clustering analysis, the conscience mechanism is only suitable for the type of sample data in which each cluster has the equal *a priori* probability, i.e., having the same number of samples. For those types of data in which each cluster has a different *a priori* probability, as discussed in [32]–[37], the competition should take those *a priori* probabilities in consideration instead of the conscience mechanism.

Second, we can observe that the DSRPCL 1 and 2 algorithms are two extreme cases of a spectrum. One implements the delearning action only to the strongest rival, in the spirit of the original RPCL learning in [14] and [15]. The other implements the delearning action to all the losers, that is, all the losers are considered as the rivals to the winner. The two kinds of delearning implementation are similar in general performance. Moreover, similar to the cases encountered by the adaptive algorithm previously proposed in [31, Sec. 6.1], we can also have a spectrum of RPCL variants by implementing the delearning action to a number of rivals between 1 and $n-1$, either deterministically or randomly. Therefore, the performances of such a spectrum of DSRPCL algorithms should be similar generally but with details spread between the DSRPCL1 and DSRPCL2 algorithms. Here, we only study the two typical algorithms.

IV. Analysis of DSRPCL Processes

As shown in the previous section, the DSRPCL algorithm is a kind of gradient-descent method on the cost function $E(\mathbf{W})$. We now analyze the convergence properties of the DSRPCL algorithm via this cost function.

A. Separation Nature

As stated in Section II, the separation nature of an RPCL process implies that a number of weight vectors will fall into a hypersphere $\mathcal{G}$ surrounding the sample data, while the other weight vectors will be driven far away from the sample data. In other words, an RPCL process separates a number of useful

weight vectors from the other extra ones in the sample space. Mathematically, a weight vector is defined to be far away from the sample data if it becomes a loser for all the samples. An RPCL process is just the sequence of the iterations of these weight vectors $\mathbf{W}^{(t)} = [W_1^{(t)}, \ldots, W_n^{(t)}]$ starting from a set of initial weight vectors $\mathbf{W} = [W_1^{(0)}, \ldots, W_n^{(0)}] \in R^{nd} - \mathcal{B}$ according to the updating rule as t increases step by step (without the stop rule). As shown by many experiments, an RPCL process always demonstrates the separation nature. Now, we prove this property for the batch DSRPCL process as follows.

Lemma 1: When a weight vector $W_i^{(t)}$ in the batch DSRPCL algorithm is far away from the sample data at time t, it will be always a loser for any sample and will go away from the sample data in the sequential iterations.

Proof: When a weight vector $W_i^{(t)}$ in the batch DSRPCL algorithm is far away from the same data, that is, it is a loser for all the samples at time t, it is on the one side of the sample data. According to (8) and (10), we have

$$\Delta W_i = \eta \sum_{\mu} \|X^{\mu} - W_i\|^{-P-2}(W_i - X^{\mu}). \qquad (16)$$

As a result, ΔW_i directs to the outside of the sample data. Therefore, $W_i^{(t+1)}$ goes more away from the sample data (i.e., each sample point).

If every weight vector goes away from the sample data, the cost function $E(\mathbf{W}^{(t)})$ will increase to infinity, which contradicts the fact that $E(\mathbf{W}^{(t)})$ decreases in the batch DSRPCL process. Thus, there exists a weight vector W_j within or around the field of the sample data at any time. As $W_i^{(t)}$ is already far away from the sample data, $W_i^{(t+1)}$ is farther away from the sample data. But W_j is close to the sample data. Then, W_j is closer to any sample than $W_i^{(t+1)}$. Therefore, $W_i^{(t+1)}$ cannot be a winner for any sample. In other words, $W_i^{(t+1)}$ is a loser for any sample. Hence, $W_i^{(t+1)}$ is farther away from the sample data and a loser for any sample.

In the same way as above for the sequential times, we have that W_i will always be a loser for any sample and will go away from the sample data in the sequential iterations. ■

Theorem 1: For a batch DSRPCL process where the learning rate η is small enough, there exists a hypersphere $\mathcal{G}$ surrounding the sample data such that after sufficient iterations, each weight vector $W_i^{(t)}$ will either: 1) fall into $\mathcal{G}$ or 2) keep outside $\mathcal{G}$ and diverge to infinity.

Proof: Because the batch DSRPCL algorithm is a gradient-descent algorithm, $E(\mathbf{W}^{(t)})$ decreases with t in the batch DSRPCL process when η is small enough. Since $E(\mathbf{W}^{(t)})$ is always positive and thus bounded below, it converges to a minimum value E^*.

While t increases to infinity, each sequence $\{W_i^{(t)}\}$ may or may not have a divergent subsequence. If $\{W_i^{(t)}\}$ has a divergent subsequence, $W_i^{(t)}$ will be far away from the sample set and will become a loser for any sample at a certain time. According to Lemma 1, it will always go away from the sample data in the sequential iterations. Therefore, it will diverge to infinity.

On the contrary, not every weight vectors will be all so far away from the sample data because $E(\mathbf{W}^{(t)})$ decreases to E^* and there must be a certain $\{W_i^{(t)}\}$ that do not have any divergent subsequence. In this case, $W_i^{(t)}$ is bounded. According to the Bolzano–Weierstrass theorem [45], there must exist certain accumulation point(s) of the subsequence(s) of the sequence $\{W_i^{(t)}\}$. Given a hypersphere $\mathcal{G}$, which surrounds all the samples as well as all the accumulation points of the nondivergent subsequences of $\{W_i^{(t)}\}$, there exists a large positive integer T such that, for $t > T$, $W_i^{(t)}$ will be always within $\mathcal{G}$.

Summing up the two results, we complete the proof. ■

According to Theorem 1, when the learning rate η is selected small enough, the separation nature holds for a batch DSRPCL process. With both the learning and delearning mechanisms in a batch DSRPCL process, some weight vectors are possible to be pushed away from the sample data. Once they go outside the hypersphere surrounding the sample data, these weight vectors will be always losers and thus delearned in subsequent iterations, such that they are driven away from the sample data and will diverge to infinity. It follows from (8) that the derivatives with respect to these weight vectors attenuate to zero. We can neglect the weight vectors that diverge to infinity since they will not affect the value of $E(\mathbf{W})$ ultimately.

On the other hand, those weight vectors that do not diverge to infinity will go towards a minimum point $\hat{\mathbf{W}}^*$ of $E(\hat{\mathbf{W}})$ in a traditional gradient-descent manner as the cost function decreases during the process, where $\hat{\mathbf{W}}$ denotes the group of weight vectors that converge. According to the differential theory, the gradient of the cost function must be zero at $\hat{\mathbf{W}}^*$. In this case, these weight vectors converge to $\hat{\mathbf{W}}^*$ even if the learning rate η is a small positive constant. Since the batch DSRPCL algorithm is a gradient-descent algorithm, these weight vectors may be stable at a saddle point of $E(\hat{\mathbf{W}})$. However, we can escape this kind of solution by disturbing the final result of the algorithm with some noise. Actually, when it is just a saddle point, the algorithm under the disturbance will leave it for a stable solution. Otherwise, the algorithm will return to the stable solution. Thus, we can neglect this case. We so regard that these weight vectors converge to a minimum point of $E(\hat{\mathbf{W}})$.

For the DSRPCL1 algorithm, the weight vectors update as each sample comes. It is an adaptive gradient-descent algorithm of the cost function. Under the mechanism of DSRPCL, two weight vectors will push each other and keep a certain distance during the learning process. Thus, when any two weight vectors are initialized with a distance, it can be considered that $\|W_i - W_j\| \geq \delta$ for any $i \neq j$, where δ is a positive constant, during a DSRPCL1 process. In this constrained case, $E(\mathbf{W})$ is twice continuously differentiable and $\{\mathbf{W} : E(\mathbf{W}) \leq C\}$ is compact for all $C > 0$. Then, $E(\mathbf{W})$ satisfies the conditions of Liung's theorem [46] and thus converges to a minimum during this adaptive learning process only if the leaning rate η attenuates in the way of (3). In the same way as Theorem 1, we can prove that a DSRPCL1 process also has the separation nature in this learning-rate setting.

For the DSRPCL2 algorithm, the adaptive gradient is actually a major part of the adaptive gradient in the DSRPCL1 algorithm. Thus, the DSRPCL2 algorithm would behave in

a way similar to the DSRPCL1 algorithm. In a DSRPCL2 process, there is a slight difference that each extra weight vector will be driven far away from the sample data and finally stop somewhere, such that it will be neither a winner nor a rival in the further competition.

In summary, a DSRPCL process has the separation nature such that a number of weight vectors converge within a hypersphere surrounding the sample data while the other weight vectors go outside the hypersphere. It can also be seen that this separation nature comes from the convergence of $E(\mathbf{W}^{(t)})$ during the iteration process.

B. Correct Division and Location

Since the separation nature really holds in a DSRPCL process, the problem of the correct convergence of the DSRPCL algorithm turns into whether a DSRPCL process has the properties of correct division and location of these converged weight vectors.

For a set of sample data with k actual clusters, the property of correct division is that k weight vectors converge within the hypersphere of the sample data, while the other $n-k$ weight vectors diverge to infinity. Moreover, the property of correct location means that the k converged weight vectors will finally locate at or around each center of the k actual clusters, respectively. Now, we analyze this issue on a set of sample data from a mixture of k separated spherical clusters of which the centers are $m_1, \ldots, m_k$, respectively. That is, each actual cluster in the sample data is the set of the samples which are closest to a vector m_i.

1) Global Minimum Point $\mathbf{W}^*$ of $E(\mathbf{W})$ at $n = k$: We consider the cost function in the decomposition $E(\mathbf{W}) = E_1(\mathbf{W}) + E_2(\mathbf{W})$ given in (7). According to its definition, $E_1(\mathbf{W})$ must have at least a minimum point, and we denote its global minimum point by $\mathbf{W}^0$. Moreover, if each actual cluster has a large number of samples, the global minimum point $\mathbf{W}^0$ is unique[1] with their component vectors located around the centers of the actual clusters $m_1, \ldots, m_k$, respectively, i.e., we approximately have $\mathbf{W}^0 = [m_1, \ldots, m_k]$.

According to (12) and (13), the sample data are divided by a given $\mathbf{W}$, into k clusters $C_1, \ldots, C_k$. On one hand, since $E_1(\mathbf{W})$ is contributed by the samples in each cluster C_i, $E_1(\mathbf{W})$ will be obviously larger than $E_1(\mathbf{W}^0)$ when $\mathbf{W}$ has a considerable distance from $\mathbf{W}^0$, that is, when the classification $\{C_1, \ldots, C_k\}$ by $\mathbf{W}$ is not consistent with the actual clusters of the mixture. On the other hand, $E_2(\mathbf{W})$ is contributed by the samples in each complementary set $\overline{C}_i$ and the minimization of $E_2(\mathbf{W})$ drives the weight vector W_i of the cluster C_i away from the samples in $\overline{C}_i$. Under an inconsistent classification, one or more actual cluster(s) may be divided into two or more parts with each of them classified to classes that are different from C_i. Correspondingly, in the complementary set of C_i, there will be a number of misclassified samples that are close to W_i. Thus, $E_2(\mathbf{W})$ will be larger than $E_2(\mathbf{W}^0)$. Summing up both conditions, it follows that $E(\mathbf{W})$ is considerably larger than $E(\mathbf{W}^0)$, that is, such a $\mathbf{W}$ cannot be the global minimum point of $E(\mathbf{W})$. Therefore, the global minimum point $\mathbf{W}^*$ of $E(\mathbf{W})$ must be close to $\mathbf{W}_0$.

Moreover, given that k weight vectors are located at each center of the k actual clusters, respectively, i.e., $\mathbf{W} = \mathbf{W}^0$, each weight vector will deviate from the center of the corresponding actual cluster as $E_2(\mathbf{W})$ further decreases from $E_2(\mathbf{W}^0)$. Then, it will cause $E_1(\mathbf{W})$ to increase. At the beginning, the decrement of $E_2(\mathbf{W})$ dominates when it is larger than the increment of $E_1(\mathbf{W})$, since the gradient of $E_1(\mathbf{W})$ around $\mathbf{W}^0$ is approximately zero. However, as $\mathbf{W}$ deviates further away from $\mathbf{W}^0$, $E_1(\mathbf{W})$ will increase quickly and thus equilibrate with the decrease of $E_2(\mathbf{W})$. As a result, the global minimum point $\mathbf{W}^*$ of $E(\mathbf{W})$ is not $\mathbf{W}^0$ but located around $\mathbf{W}^0$.

2) Deviation of $\mathbf{W}^*$ From $\mathbf{W}^0$: From the expression of $E_2(\mathbf{W})$, we can see that the deviation of $\mathbf{W}^*$ from $\mathbf{W}^0$, i.e., the distance between $\mathbf{W}^*$ and $\mathbf{W}^0$, depends on the style of delearning and the selection of P. If the delearning is implemented on the rival only, the decrement of $E_2(\mathbf{W})$ is small. Thus, the deviation is small. If the delearning is implemented on all the losers, the decrement is obviously increased; thus, the deviation becomes larger.

On the other hand, as P becomes larger, the decrement becomes lower. Thus, the deviation becomes smaller. Otherwise, the deviation becomes larger. Therefore, if P is selected large enough, the deviation can be controlled in a small neighborhood of the origin. That is, $\mathbf{W}^*$ is closely around $\mathbf{W}^0$. In other words, the k weight vectors of $\mathbf{W}^*$ are located closely around each center of the k actual clusters, respectively.

3) Variation of the Global Minimum of $E(\mathbf{W})$ With n: Since $E_1(\mathbf{W}) = E_{\mathrm{MSE}}(\mathbf{W})$, its global minimum value decreases with n, which is shown in Fig. 1(a). On the other hand, as n increases, the number of the samples in each complementary set $\overline{C}_i$ increases and thus $E_2(\mathbf{W})$ increases. Therefore, the global minimum value of $E_2(\mathbf{W})$ increases with n, which is shown in Fig. 1(b).

Adding the two curves of the variation of the global minimum value of $E_1(\mathbf{W})$ and $E_2(\mathbf{W})$ together, we have the curve of the variation of the global minimum of $E(\mathbf{W})$, which is shown in Fig. 1(c). The global minimum value of $E(\mathbf{W})$ is just located at $n = k$. In fact, as n decreases from k, the best classification based on $\mathbf{W}$ cannot match the structure of the actual clusters in the mixture. In this case, $E_1(\mathbf{W})$ increases considerably, while $E_2(\mathbf{W})$ decreases at a low degree. Therefore, the global minimum of $E(\mathbf{W})$ still decreases with n when $n \leq k$. However, as n increases from k, the best classification can match the structure of the clusters in the mixture in such a way that an actual cluster is divided into two smaller clusters. In this case, $E_1(\mathbf{W})$ decreases at a low degree, while $E_2(\mathbf{W})$ increases considerably. Therefore, the global minimum value of $E(\mathbf{W})$ increases with n when $n \geq k$. As a result, the curve of the variation of the global minimum value of $E(\mathbf{W})$ is convex, and the minimum point is $n = k$.

Furthermore, if the $n-k$ extra weight vectors are put to infinity and do not affect $E(\mathbf{W})$ anymore, the global minimum value of $E(\mathbf{W})$ does not increase from k, which is shown in Fig. 1(d).

[1]Here, we do not consider the interchanging of weight-vector labels or the columns in $\mathbf{W}$.

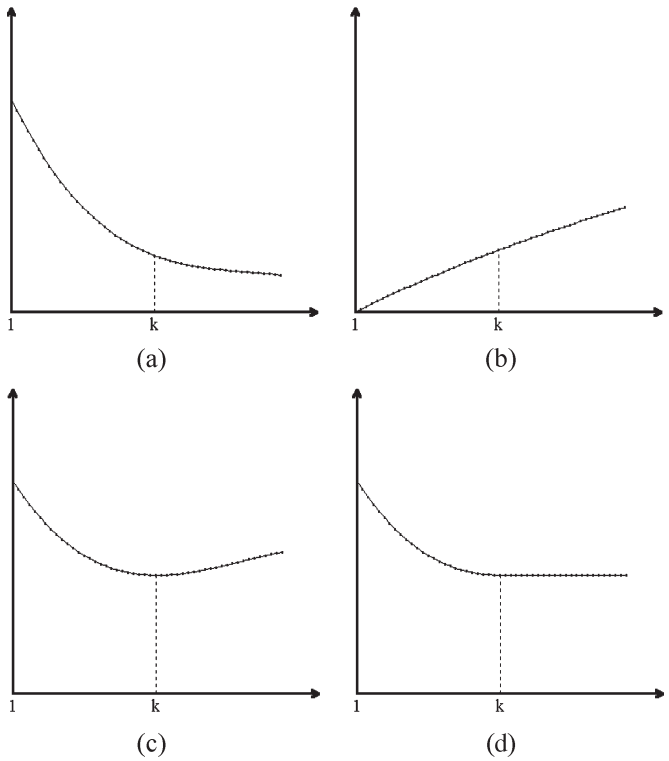


Fig. 1. Sketches of the global minimum value of the cost functions with respect to n, where k is the correct number of the clusters in the sample data. (a) Sketch of the global minimum value of $E_1(\mathbf{W})$. (b) Sketch of the global minimum value of $E_2(\mathbf{W})$. (c) Sketch of the global minimum value of $E(\mathbf{W})$. (d) Sketch of the global minimum value of $E(\mathbf{W})$ with the extra weight vectors set to infinity.

Summing up the above analysis, we observe that the global minimum point of $E(\mathbf{W})$ is unique and corresponds to the best classification on the sample data.[2] In other words, a DSRPCL process has the properties of correct division and location as long as $E(\mathbf{W}^{(t)})$ converges to the global minimum of $E(\mathbf{W})$. Although the above heuristic analyses are not so strict to be as a mathematical proof, they are intuitive and reasonable. Furthermore, as will be shown in the next section, they are consistent with the empirical results that demonstrate that the global minimum of $E(\mathbf{W})$ arrives at $n = k$ with the correct number of weight vectors located around each center of the actual clusters, respectively, but with the extra weight vectors driven far away from the sample data.

In addition, as a type of descent method, the DSRPCL algorithm may be trapped in a local minimum. However, we can implement the DSRPCL algorithm together with a mechanism of searching the global minimum of $E(\mathbf{W})$. The simulated-annealing technique will be applied to the DSRPCL algorithm on the simulation experiments in the next section.

C. Selection of P

According to the previous analysis, the global minimization of $E(\mathbf{W})$ provides us a new criterion to determine the number clusters in a sample data set. That is, if $E(\mathbf{W})$ based on a sample data set is globally minimized at $n = k$, the number of clusters in the sample data set should be k. However, it has been found from the simulation and experimental results that P in the cost function or the DSRPCL algorithm should be selected properly; otherwise, this new criterion or the DSRPCL algorithm will not work efficiently. In fact, if P is too small, the power of delearning may become so strong that the majority of the weight vectors will be pushed away from the sample data; conversely, if P is too large, the power of delearning may become so weak that only a few or none of the weight vectors will be pushed away from the sample data. In other words, a smaller value of P tends to make the criterion select a smaller number of clusters in a sample data set, while a larger value of P tends to make the criterion select a larger number of clusters in a sample data set. However, it is shown by the experiments that there exists a wide interval $[P_0, P_1]$ of P such that the new criterion can be reasonable and consistent with the number of actual clusters in the sample data set. In many cases of the sample data, we have found that only if $P \geq 0.01$, the global minimum point of $E(\mathbf{W})$ is close to the global minimum point of $E_1(\mathbf{W})$ (at $n = k$), which corresponds to the correct division and location of the weight vectors on the sample data. Thus, in order to make the converged weight vectors be located around each center of the clusters in the sample data, respectively, P_0 should not be smaller than 0.01. On the other hand, it has been shown by experiment that if $P > 1.9$, some extra weight vectors cannot be pushed away from the sample data in certain cases. Thus, P_1 should not be lager than 1.9. Experimentally, as P is selected around 0.2, the DSRPCL algorithm performs well. Hence, P will always be selected by 0.2 in our experiments in the next section.

V. Experimental Results

In this section, several simulation experiments are carried out to demonstrate our proposed cost function as well as the DSRPCL algorithms. Moreover, the DSRPCL algorithms are applied to unsupervised color image segmentation and classification of the wine data.

A. Simulation Experiments for Clustering Analysis

1) Sample Data Sets: We begin with a description of the five sets of sample data used for our simulation experiments. We conducted five Monte Carlo experiments where samples were drawn from a mixture of four or three bivariate Gaussian distributions on the plane coordinate system (i.e., $d = 2$). All the Gaussian distributions are cap-shaped, that is, their covariance matrices have the form of $\sigma^2 I$, where σ is the standard variance.

As shown in Fig. 2, the five sets of sample data are generated at different degrees of overlap among the clusters in the mixture by controlling the variances of Gaussian distributions and with equal or unequal mixing proportions of the clusters in the mixture by controlling the number of samples in each cluster. The detailed parameters for these five sets of sample data are given in Table I, where m_i, σ_i, α_i, and N_i denote the mean vector, the standard variance, the mixing proportion, and the number of samples of the ith Gaussian, respectively.

[2]Note that the extra weight vectors must be set to infinity.

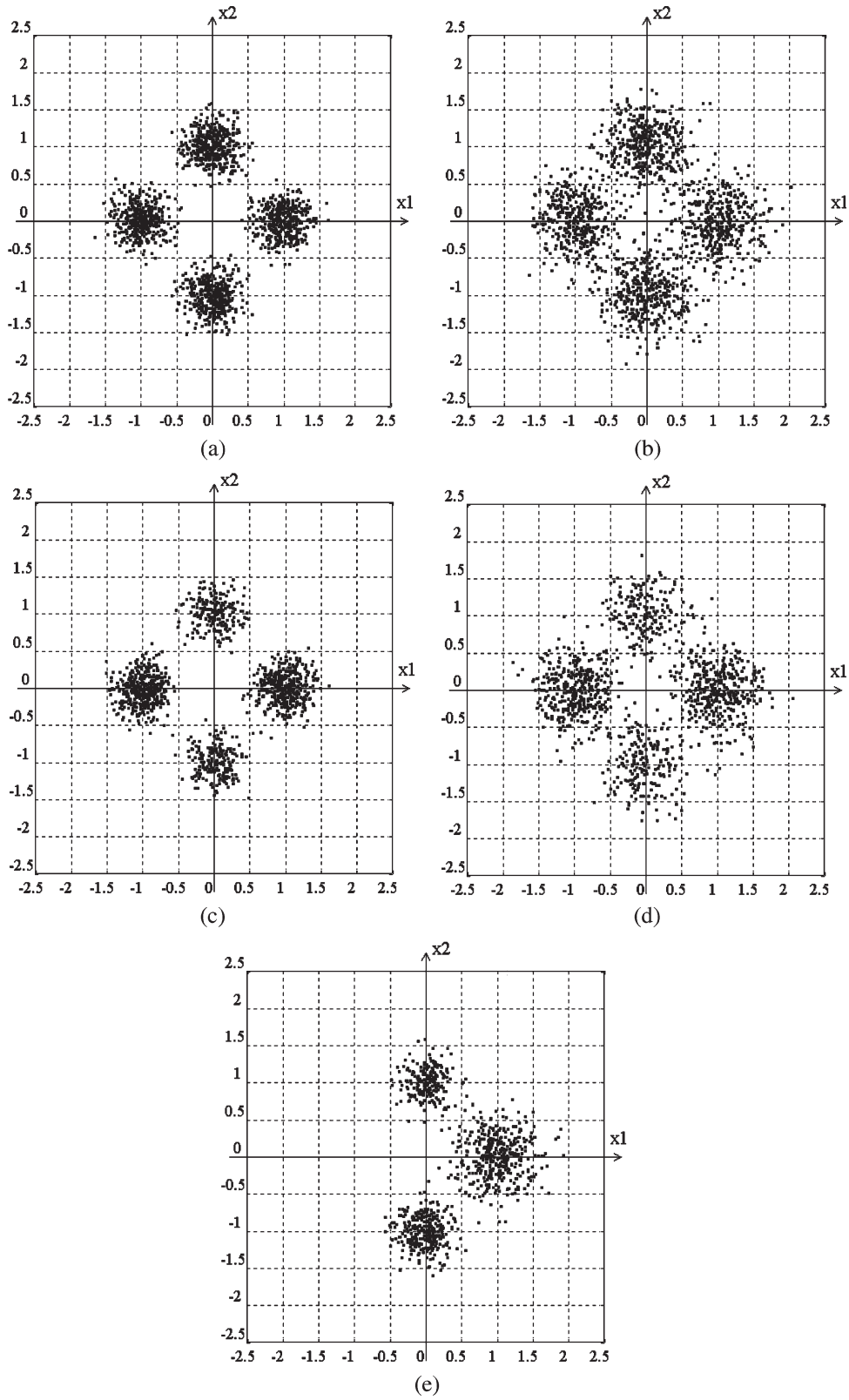


Fig. 2. Five sets of sample data used in the experiments. (a) First set of the sample data $\mathcal{S}_1$. (b) Second set of the sample data $\mathcal{S}_2$. (c) Third set of the sample data $\mathcal{S}_3$. (d) Fourth set of the sample data $\mathcal{S}_4$. (e) Fifth set of the sample data $\mathcal{S}_5$.

2) Simulation Results of the DSRPCL Algorithms: According to the analysis on the cost function, the correct convergence of the DSRPCL algorithm is equivalent to that $E(\mathbf{W}^{(t)})$ converges to the global minimum of $E(\mathbf{W})$ when n is greater than the number of the actual clusters in the sample data. If the minimum point of $E(\mathbf{W})$ is unique and global on a set of sample data, the DSRPCL algorithm will always converge correctly. In fact, when the set of sample data has a symmetric structure like

TABLE I
PARAMETERS OF THE FIVE SETS OF SAMPLE DATA

The Sample Set	Cluster	m_i	σ_i	α_i	N_i
$\mathcal{S}_1$ ($N = 1600$)	Cluster 1	(-1, 0)	0.2	0.250	400
	Cluster 2	(1, 0)	0.2	0.250	400
	Cluster 3	(0, 1)	0.2	0.250	400
	Cluster 4	(0, -1)	0.2	0.250	400
$\mathcal{S}_2$ ($N = 1600$)	Cluster 1	(-1, 0)	0.3	0.250	400
	Cluster 2	(1, 0)	0.3	0.250	400
	Cluster 3	(0, 1)	0.3	0.250	400
	Cluster 4	(0, -1)	0.3	0.250	400
$\mathcal{S}_3$ ($N = 1200$)	Cluster 1	(-1, 0)	0.2	0.333	400
	Cluster 2	(1, 0)	0.2	0.333	400
	Cluster 3	(0, 1)	0.2	0.166	200
	Cluster 4	(0, -1)	0.2	0.166	200
$\mathcal{S}_4$ ($N = 1200$)	Cluster 1	(-1, 0)	0.3	0.333	400
	Cluster 2	(1, 0)	0.3	0.333	400
	Cluster 3	(0, 1)	0.3	0.166	200
	Cluster 4	(0, -1)	0.3	0.166	200
$\mathcal{S}_5$ ($N = 900$)	Cluster 1	(1, 0)	0.3	0.444	400
	Cluster 2	(0, 1)	0.2	0.333	200
	Cluster 3	(0, -1)	0.2	0.222	300

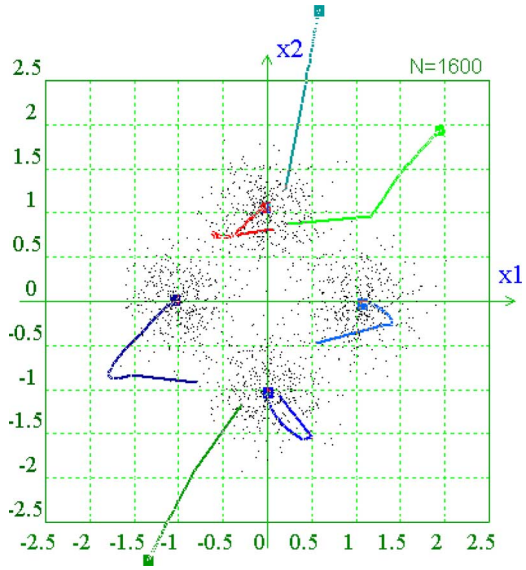


Fig. 3. Trajectories of weight vectors (solid blocks) during the learning process of the batch DSRPCL algorithm on the second set of sample data in the case $n = 7$ and $k = 4$. The learning rate η was selected to be 0.001. The number of iterations is 31. (Color version available online at http://ieeexplore.ieee.org.)

$\mathcal{S}_1$ and $\mathcal{S}_2$, i.e., the number of the samples in each cluster is the same and these clusters are separated at a certain degree; it is usual that the minimum point of $E(\mathbf{W})$ is unique and global. Thus, the DSRPCL algorithm will converge correctly on this kind of sample data. In order to demonstrate this result, we run the batch DSRPCL, DSRPCL1, and DSRPCL2 algorithms on the second set of sample data $\mathcal{S}_2$. Since there exists some deviation between the global minimum of the cost function and the true parameters and this deviation is caused by $E_2(\mathbf{W})$, we can make the leaning rate concerning the derivatives of $E_2(\mathbf{W})$ in the DSRPCL algorithm (i.e., the delearning rate) attenuate to zero so that the deviation will be eliminated eventually. In our experiments here or in the following sections, we add this deviation-eliminating mechanism to the DSRPCL algorithm. For the three DSRPCL algorithms, we set the delearning rate to be η/m, where $m = [t/5]$ for the batch DSRPCL algorithm

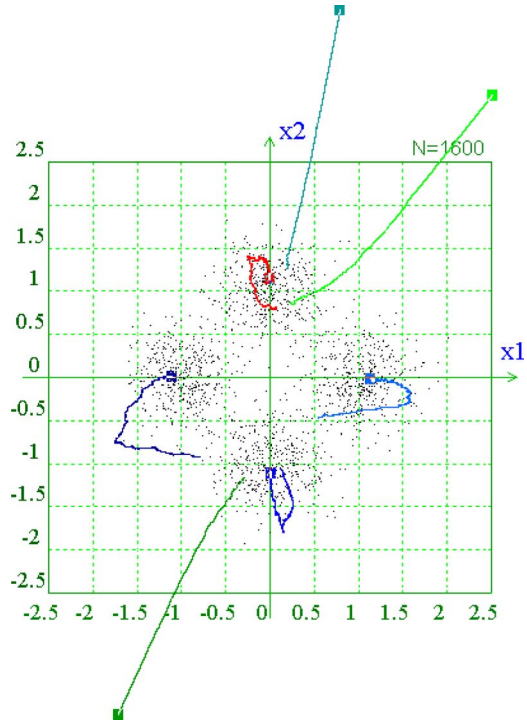


Fig. 4. Trajectories of weight vectors (solid blocks) during the learning process of the DSRPCL1 algorithm on the second set of sample data in the case $n = 7$ and $k = 4$. The learning rate η was selected to be 0.003. The number of iterations is 16 018. (Color version available online at http://ieeexplore.ieee.org.)

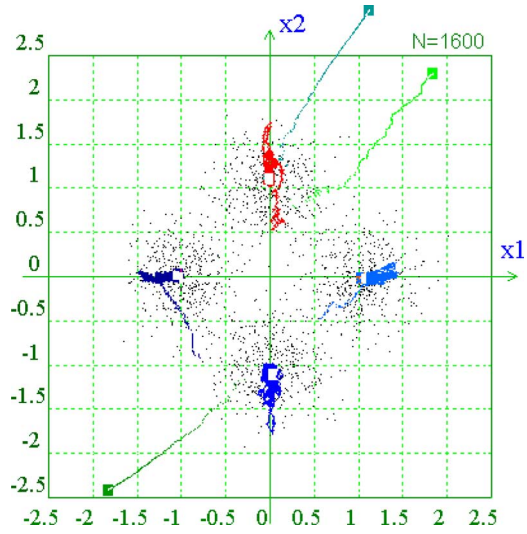


Fig. 5. Trajectories of weight vectors (solid blocks) during the learning process of the DSRPCL2 algorithm on the second set of sample data in the case $n = 7$ and $k = 4$. The learning rate η was selected to be 0.02. The number of iterations is 16 023. (Color version available online at http://ieeexplore.ieee.org.)

and $m = [t/5N]$ for the DSRPCL1 and DSRPCL2 algorithms, where $[x]$ is the upper integer of a real number x, t is the time, i.e., the number of updated iterations, and N is the number of samples.

The experimental results of the batch DSRPCL, DSRPCL1, and DSRPCL2 algorithms in the case of $n = 7$ and $k = 4$, are given in Figs. 3–5, respectively. The learning rates of the three

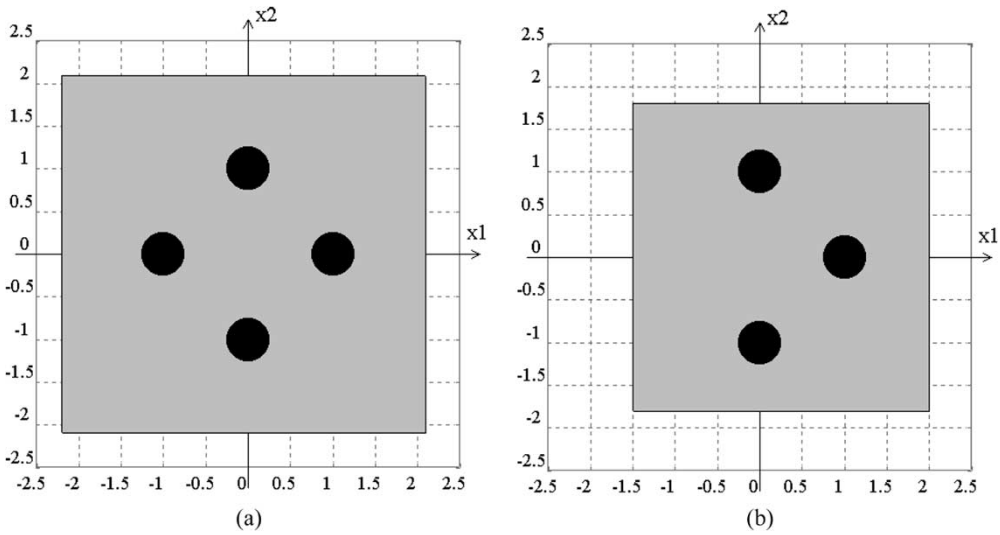


Fig. 6. Regions for checking the validity of the empirical results. There should be one weight vector in each black region and no weight vectors in the gray region. (a) In the case of the first to fourth sample data, the four black regions are the circles of the radius 0.25 at $(-1, 0)$, $(1, 0)$, $(0, 1)$, and $(0, -1)$, respectively. The gray region is the difference set of the envelope $[-2.1, 2.1] \times [-2.1, 2.1]$ of the sample data to the union of the four black circles. (b) In the case of the fifth sample data, the three black regions are the circles of the radius 0.25 at $(1, 0)$, $(0, 1)$, and $(0, -1)$, respectively. The gray region is the difference set of the envelope $[-1.5, 2.0] \times [-1.8, 1.8]$ of the sample data to the union of the three black circles.

algorithms are selected to be 0.001, 0.003, and 0.02, respectively. We stop each algorithm when $\Delta E_t(\mathbf{W}) = |E(\mathbf{W}^{(t)}) - E(\mathbf{W}^{(t-1)})| < 10^{-6}$. From the three figures, we observe that all the three DSRPCL algorithms result in the correct convergence. That is, the four weight vectors are finally located around the centers of the four actual clusters, respectively, while the three extra weight vectors are driven away from the sample data. It is clear that the deviation between each converged weight vector and the corresponding center of the actual cluster is almost eliminated with the deviation-eliminating mechanism.

On the other hand, we have found that the DSRPCL algorithm may result in the wrong convergence when n becomes larger. In this case, the DSRPCL algorithm is certainly trapped in a local minimum of $E(\mathbf{W})$ when the resulted classification is wrong. Therefore, the uniqueness of the minimum point of the cost function $E(\mathbf{W})$ is broken at such n, though such phenomenon seldom appears if $n \leq 2k$.

3) Simulation Results of the Simulated-Annealing RPCL (SARPCL) Algorithm: For a set of sample data with an asymmetric structure like $\mathcal{S}_3$, $\mathcal{S}_4$, and $\mathcal{S}_5$, the DSRPCL algorithm does converge incorrectly even if n is slightly larger than k. As n becomes larger, this phenomenon appears more frequently. That is, there exist a number of local minimums and saddle points. In order to avoid or eliminate this phenomenon, we can utilize some global searching techniques such as simulated annealing [7], genetic algorithms [47], and evolution computation [48]. Here, we apply the simulated-annealing mechanism to the DSRPCL1 algorithm, resulting in the following SARPCL algorithm.

SARPCL Algorithm

Step 1) Randomly and independently select each initial weight w_{ij} from the interval $[a, b]$, and let $T = 0$.

Step 2) At timestep T, let $\lambda = \exp(-k_1 T - k_0)$, $\eta = \eta_0/(c_1 T + c_0)$, and $t = 0$.

Step 3) At subtimestep t, randomly select a sample pattern X^μ from $\mathcal{S} = \{X^1, X^2, \ldots, X^N\}$ and take a random number ξ subject to the uniform distribution on [0, 1]. If $\xi > \lambda$, the weight vectors are modified by

$$\Delta W_i = \begin{cases} \eta(X^\mu - W_i), & \text{if } i = c(\mu) \\ -\eta \|X^\mu - W_i\|^{-P-2}(X^\mu - W_i), & \text{otherwise.} \end{cases} \quad (17)$$

Otherwise, if $\xi \leq \lambda$, the weight vectors are modified by

$$\Delta W_i = \begin{cases} -\eta(X^\mu - W_i), & \text{if } i = c(\mu) \\ \eta \|X^\mu - W_i\|^{-P-2}(X^\mu - W_i), & \text{otherwise.} \end{cases} \quad (18)$$

Step 4) If $t < M$, we let $t = t + 1$ and return to Step 3).

Step 5) If $\lambda < \varepsilon$, stop; otherwise, let $T = T + 1$ and return to Step 2).

In the SARPCL algorithm, the delearning rate is set to be $\eta/(T + 1)$. Actually, we can consider $1/T$ as the temperature in the process of an annealing simulation. On each stage of a temperature point, the weight vectors will evolve for M times to arrive at an equilibrium distribution of the weight vectors. Thus, M is related to n. The learning rate η is generally selected by experience. The stopping threshold value ε is usually selected to be 10^{-6}. The other way to stop the algorithm is to set an upper threshold value for T. Moreover, k_0, k_1, c_0, and c_1 are positive numbers, which are also selected by experience, and $[a, b]$ is the interval for the initial weights.

In the experiments of the SARPCL algorithm on the five sets of sample data, the parameters are selected as follows: $\eta_0 = 0.003$, $M = 100$, $k_1 = 0.005$, $k_0 = 1.200$, $c_1 = 0.015$, $c_0 = 1.000$, and $[a, b] = [-1.2, 1.2]$. We stop the algorithm and get the empirical results when $T = 10000$.

For checking the validity of an empirical result of the SARPCL algorithm, i.e., a group of resulted weight vectors, on a set of sample data, we adopt a group of checking regions

TABLE II
VALID RANGES OF n ON THE FIVE SETS OF SAMPLE DATA

The set of sample data	$\mathcal{S}_1$	$\mathcal{S}_2$	$\mathcal{S}_3$	$\mathcal{S}_4$	$\mathcal{S}_5$
VP = 100%	4-36	4-9	4-8	4-6	3-5
VP $\geq$ 97%	4-62	4-21	4-15	4-9	3-9
VP $\geq$ 95%	4-65	4-38	4-25	4-14	3-10

as shown in Fig. 6. The procedure aims to check whether the SARPCL algorithm converges correctly or not. Being combined with the annealing-simulation mechanism, the SARPCL algorithm makes $E(\mathbf{W}^{(t)})$ converge to the global minimum of $E(\mathbf{W})$ more probably. According to the analysis on the cost function, this means that each group of the resulted weight vectors is valid with a high probability, demonstrated by the simulation experiments when n is not much larger than k. However, as n becomes much larger than k, the local minimum points of $E(\mathbf{W})$ may increase rapidly such that a group of resulted weight vectors will be invalid with a considerable large probability, even if the SARPCL algorithm is used. Thus, the usefulness of the SARPCL algorithm relies on whether there exists a large valid range of n in which the SARPCL algorithm converges correctly with a probability near 1. By the following simulation results, we will demonstrate that there indeed exists a large valid range of n for each experimental sample data set.

On each data set, we run the SARPCL algorithm for 100 times at each value of n ($\geq k$) from k. In order to determine the valid range of n, we increase n from k and compute at each n the percentage of the valid empirical result, i.e., the number of the valid results over 100. The upper bound of the valid range can be estimated by solving the largest integer of n at which the valid percentage is larger than or equal to a threshold value near one. We set this threshold value to be 1, 0.97, and 0.95, respectively, and get the valid ranges on the five sets of sample data listed in Table II.

In Table II, VP represents the valid percentage of the empirical results of the SARPCL algorithm. When the set of sample data has a symmetric structure such as $\mathcal{S}_1$ and $\mathcal{S}_2$, the valid range is rather large; otherwise, if the set of sample data has an asymmetric and complex structure, the valid range is relatively small. After all, the valid ranges for the five sets of sample data are reasonably large for applying the SARPCL algorithm to an unsupervised classification.

Based on all these simulation experiments, we can find that as the cost reduces into the global minimum through a DSRPCL process, a correct number of weight vectors is automatically selected and located around the centers of the clusters, respectively. That is, this cost function can both detect the number of actual clusters and make a reasonable classification on such a data set via its minimization through the DSRPCL or the ASRPCL algorithm. However, if two or more clusters are seriously overlapped, the DSRPCL algorithm generally regards them as one cluster and leads to a wrong result, which is also demonstrated by the simulation experiments.

From the simulation experiments of both the DSRPCL and the ASRPCL algorithms, we can find that when each initial weight w_{ij} is randomly and independently selected from an interval $[a, b]$ such that each initial weight vector is randomly selected in the field or the hypersphere of the sample data, the dead unit problem is avoided completely. Thus, the DSRPCL algorithm can overcome the dead unit problem without the conscience mechanism only if the initial weight vectors are selected appropriately. Moreover, as there is no constraint that all the winning frequencies are equal in the DSRPCL algorithm, the classification accuracy of the sample data in which each cluster has a different *a priori* probability is improved considerably in comparison with that of the original RPCL algorithm.

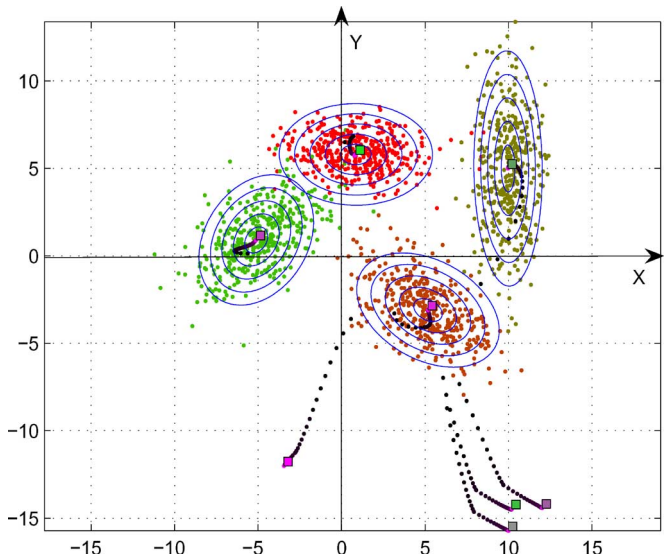


Fig. 7. Trajectories of weight vectors (solid blocks) during the learning process of the batch DSRPCL algorithm using the Mahalanobis distance on a sample data set consisting of four elliptical clusters or Gaussians in the case $n = 8$ and $k = 4$. The learning rate η was selected to be 0.001. The number of iterations is 100. In this figure, the contour lines of each Gaussian based on a pair of final estimated mean vector and covariance matrix obtained from the DSRPCL algorithm are retained unless its density is less than e^{-3} (peak). (Color version available online at http://ieeexplore.ieee.org.)

By the other experiments, we further find that the DSRPCL algorithm works well on the sample data set with a large number clusters (e.g., 40 clusters) or in a higher dimensional space (e.g., 20-dimensional space) only if these clusters are separated at a degree as those in the above sample data sets. When the clusters become elliptical or the other forms like a triangle or a rectangle, the DSRPCL algorithm can still detect the number of clusters in the sample data only if these clusters are separated at a similar degree. However, the classification may be not so good. Actually, for the case of elliptical clusters, we can use the Mahalanobis distance instead of the Euclidean distance in the cost function to improve the classification performance. In this way, the DSRPCL algorithm changes greatly according to the new derivatives of the cost function with respect to the mean vectors and covariance matrixes, which are omitted in this paper. However, a typical experimental result of the DSRPCL algorithm using the Mahalanobis distance is given Fig. 7, from which we can find that the Mahalanobis distance-based DSRPCL algorithm can solve the classification problem of elliptical clusters efficiently.

B. Experiments for Unsupervised Color Image Segmentation

Segmenting a digital color image into homogenous regions corresponding to the objects (including the background) is a

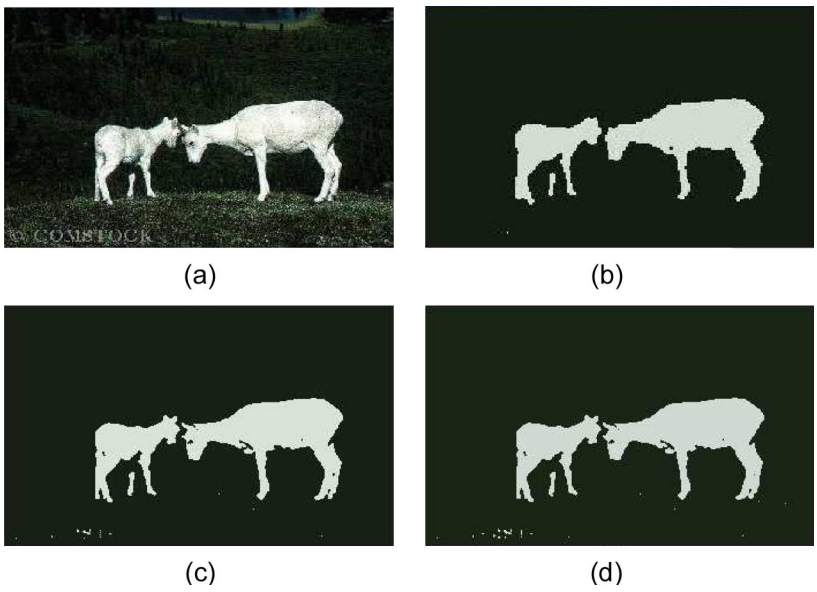


Fig. 8. Experimental results on the color image of two goats. (a) Original color image. (b) Segmentation image of the GCC algorithm. (c) Segmented image of the DSRPCL1 algorithm after 75 488 iterations (the execution time on a Pentium-4 personal computer was 14.641 s). (d) Segmented image of the DSRPCL2 algorithm after 75 645 iterations (the execution time on a Pentium-4 personal computer was 7.938 s). (Color version available online at http://ieeexplore.ieee.org.)

fundamental problem in image processing. When the number of objects in an image is not known in advance, the image segmentation is in an unsupervised mode and becomes rather difficult in practice [49]. However, the DSRPCL algorithm provides a new tool for solving this unsupervised-color-image-segmentation problem. We apply it to the unsupervised color image segmentation on two typical color images that are expressed in the three-dimensional color space by the RGB system. Actually, we use each weight vector in the DSRPCL algorithm to represent an object in a color image and set k to be larger than the number k^* of the actual objects in the image. When the weight vectors are far from the color representation field of the image, we cancel these weight vectors. Finally, the pixels in the image are partitioned according to the converged weight vectors under the minimum distance principle.

In our experiments, for ease of segmentation, we make a median filtering on each of the two regularized images and regularize all the three coordinates of the pixels in each color image by dividing them by 32 so that the regularized coordinates are within an interval of [0, 8]. After such a preprocessing, we run the DSRPCL algorithm on the data sets of the two color images, respectively, by letting $k = 6$ with the stopping criterion as above. The initial parameters are randomly selected in some intervals. Since the batch DSRPCL algorithm has the similar convergence behavior as the DSRPCL1 algorithm, we only apply the DSRPCL 1 and 2 algorithms to the unsupervised color image segmentation. Here, the learning rates of the DSRPCL 1 and 2 algorithms are selected to be 0.1 and 0.5, respectively.

The experimental results of the two DSRPCL algorithms on the color image of two goats are given in Fig. 8. As compared with the original image given in Fig. 8(a), the unsupervised-color-image-segmentation results of the two DSRPCL algorithms are rather good. From the two segmented images given by Fig. 8(c) and (d), we can observe that two objects are finally located accurately. That is, the partitions accurately match the actual objects in the image. Although k is set to be six for both DSRPCL algorithms, they only find two objects in the image, i.e., keep two weight vectors in the color representation field, with the other weight vectors being driven far away from the color representation field. Moreover, the experimental results of the two DSRPCL algorithms on the color image of an elephant are given in Fig. 9. From the two segmented images given by Fig. 9(c) and (d), we can also observe that three objects are finally located accurately. As a result, the DSRPCL algorithm can be successfully applied to the unsupervised color image segmentation.

Finally, we compare the DSRPCL algorithm with the original RPCL algorithm and the generalized competitive clustering (GCC) algorithm proposed in [49] based on our unsupervised-color-image-segmentation experiments as well as the simulated experiments on the randomly generated data sets from Gaussian mixtures as above. In comparison with the original RPCL algorithm [14], [15], the DSRPCL algorithm owns a better convergence behavior. Generally, the DSRPCL or SARPCL algorithm is not so sensitive to the initial values of the weight vectors and leads to a good result. Oppositely, the original RPCL algorithm is sensitive to the initial values of the weight vectors, and the conscience mechanism makes it difficult to locate the clusters or objects accurately. Moreover, the segmentation results of the DSRPCL algorithm are also better than those of the GCC algorithm based on the fuzzy theory given in the Web (http://www-rocq.inria.fr/~boujemaa/Partielle2.html). For comparison, we give the segmentation results of the GCC algorithm on these two images by Figs. 8(b) and 9(b), respectively, downloaded from the above Web site. By comparing the segmentation results of the DSRPCL and GCC algorithms on these two color images, we can find that the DSRPCL algorithms lead to a more accurate segmentation on the contours of the objects in each image. By our experiments, we even find that the execution time of the DSRPCL algorithm is generally shorter than that

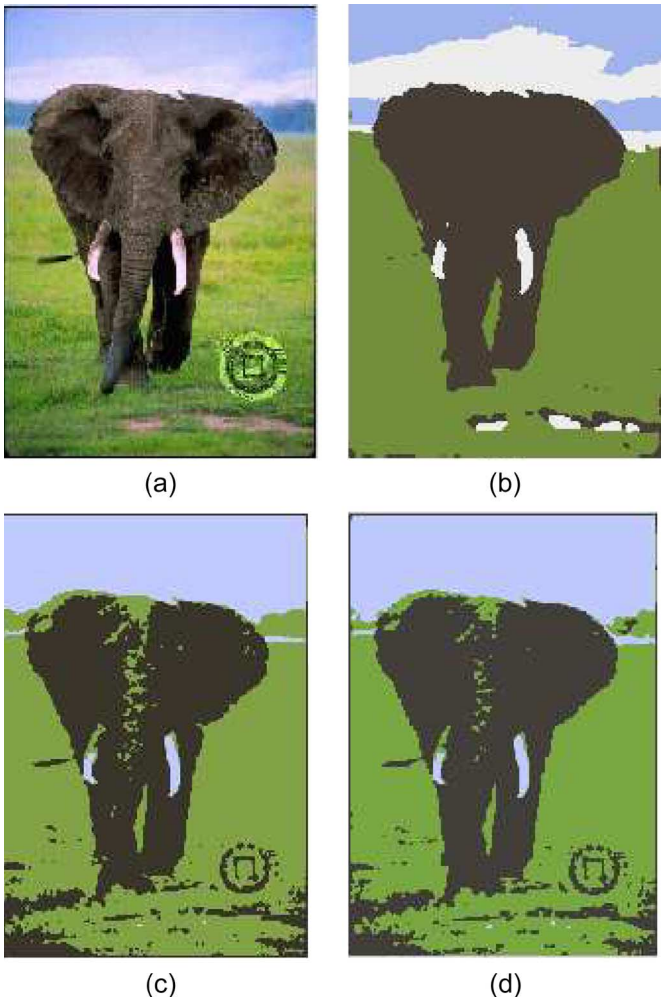


Fig. 9. Experimental results on the color image of an elephant. (a) Original color image. (b) Segmentation image of the GCC algorithm. (c) Segmented image of the DSRPCL1 algorithm after 73 024 iterations (the execution time on a Pentium-4 personal computer was 15.063 s). (d) Segmented image of the DSRPCL2 algorithm after 73 843 iterations (the execution time on a Pentium-4 personal computer was 7.875 s). (Color version available online at http://ieeexplore.ieee.org.)

of the GCC algorithm. As a whole, the DSRPCL algorithm is an efficient algorithm for clustering analysis and unsupervised color image segmentation.

C. Classification of the Wine Data

We further apply the DSRPCL algorithm to classification (or recognition) of the wine data [50], which are typical real data for testing the classification algorithm. Actually, the wine dataset consists of 178 samples of three types of wine. Each datum is 13-dimensional and consists of chemical analyses of a sample from a certain type of wine. We first regularize these wine data into an interval of [0, 8] and then apply the SARPCL algorithm to solving the unsupervised classification problem of the wine data by setting $k = 6$ with the parameters being selected similarly as above. It is shown by the experiments that the SARPCL algorithm can detect the three classes in the wine dataset with an optimal classification accuracy of 97.20% (there are five errors), which is slightly less than the classification accuracy 97.75% (there are four errors) of the method of linear mixing kernels (Gaussians) with the information minimization criterion [51]. Moreover, when the Mahalanobis distance is used in the SARPCL algorithm, the best classification accuracy on the wine data set can reach to 100%, i.e., all the wine samples can be classified correctly.

D. Discussions on the Behaviors of the DSRPCL Algorithms

Finally, we present some discussions on the behaviors of the DSRPCL algorithms. As shown by the simulation and application experiments, the DSRPCL algorithms can allocate a correct number of weight vectors at or around the centers of the actual clusters, respectively, with the extra weight vectors being driven outside the hypersphere surrounding the sample data. That is, the DSRPCL algorithms converge similarly as the original RPCL algorithm. However, they have certain different behavior characteristics. The batch DSRPCL algorithm converges more stably than the DSRPCL 1 and 2 algorithms, but these two adaptive DSRPCL algorithms converge much more quickly than batch one. Moreover, they are not so sensitive to the initial values of the weight vectors as batch one. Clearly, the sensitivity of initial values of weight vectors is improved considerably by the SARPCL algorithm, but the simulated-annealing procedure incurs a larger computational cost.

As for the two adaptive DSRPCL algorithms, the first algorithm can quickly drive the extra weight vectors far away from the sample data set, while the second algorithm can drive the extra weight vectors just in the outside of the sample data. For the sample data set in which the actual clusters are well separated, the two algorithms can both converge to a good result. But when some actual clusters are overlapped at a certain degree, the first algorithm tends to drive more weight vectors than the second algorithm, which is demonstrated by some experiments on unsupervised color image segmentation. The reason is just that the first algorithm implements the delearning action to all losers, while the second algorithm implements the delearning action only to the strongest loser, i.e., the rival. As pointed out at the end of Section III, these two algorithms are the two extremes of DSRPCL. It is also found by the experiments that the intermediate schemes of DSRPCL usually lead to a result between those of the two extremes on the determination of the number of clusters. However, in certain cases of the sample data, they can even outperform both DSRPCL 1 and 2 algorithms on the classification accuracy.

VI. Conclusion

A cost-function approach is proposed to investigate the problem of RPCL correct convergence via a DSRPCL algorithm that is associated with the minimization of a cost function on the weight vectors. As the cost function reduces to a local minimum, the DSRPCL algorithm has the separation nature that a number of weight vectors converge within a hypersphere surrounding the sample data, while the other weight vectors diverge to infinity. If we make the cost function reduce to the global minimum, the DSRPCL algorithm will allocate a correct

number of weight vectors that will converge to each center of the clusters in the sample data, respectively.

ACKNOWLEDGMENT

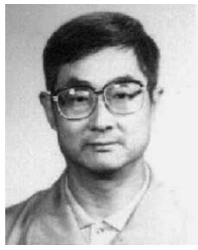
The authors would like to thank the editor and three anonymous reviewers for their valuable comments and insightful suggestions; L. Xu for his strong support, valuable suggestions, and helpful discussions on this research; and B. Cao for his support of the experiments.

REFERENCES

[1] P. A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*. London, U.K.: Prentice-Hall, 1982.
[2] J. Makhoul, S. Rpucos, and H. Gish, "Vector quantization in speech coding," *Proc. IEEE*, vol. 73, no. 11, pp. 1551–1558, Nov. 1985.
[3] N. Nasrabadi and R. A. King, "Image coding using vector quantization: A review," *IEEE Trans. Commun.*, vol. 36, no. 8, pp. 957–971, Aug. 1988.
[4] J. Moody and C. Darken, "Fast learning in networks of locally tuned processing units," *Neural Comput.*, vol. 1, no. 2, pp. 281–294, 1989.
[5] T. Poggio and F. Girosi, "Regularization algorithms for learning that are equivalent to multilayer networks," *Science*, vol. 247, no. 4945, pp. 978–982, 1990.
[6] S. J. Nowlan and G. E. Hinton, "Evaluation of adaptive mixtures of competing experts," in *Advances in Neural Information Processing System*, vol. 3, R. P. Lippmann, J. E. Moody, and D. S. Touretzky, Eds. San Mateo, CA: Morgan Kaufmann, 1991, pp. 774–780.
[7] R. Hecht-Nielsen, *Neurocomputing*. Reading, MA: Addison-Wesley, 1990.
[8] D. E. Rumelhart and D. Zipser, "Feature discovery by competitive learning," *Cogn. Sci.*, vol. 9, no. 1, pp. 75–112, 1985.
[9] S. Grossberg, "Competitive learning: From iterative activation to adaptive resonance," *Cogn. Sci.*, vol. 11, no. 1, pp. 23–83, 1987.
[10] R. Hecht-Nielsen, "Counterpropagation networks," *Appl. Opt.*, vol. 26, no. 23, pp. 4979–4984, Dec. 1987.
[11] E. L. Bienenstock, L. N. Cooper, and P. W. Munro, "Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex," *J. Neurosci.*, vol. 2, no. 1, pp. 32–48, Jan. 1982.
[12] D. DeSieno, "Adding a conscience to competitive learning," in *Proc. IEEE Int. Conf. Neural Networks*, San Diego, CA, 1988, vol. 1, pp. 117–124.
[13] S. C. Ahalt, A. K. Krishnamurty, P. Chen, and D. E. Melton, "Competitive learning algorithms for vector quantization," *Neural Netw.*, vol. 3, no. 3, pp. 277–291, 1990.
[14] L. Xu, A. Krzyzak, and E. Oja, "Unsupervised and supervised classification by rival penalized competitive learning," in *Proc. 11th Int. Conf. Pattern Recognition*, The Hague, The Netherlands, Aug. 30–Sep. 3 1992, vol. 1, pp. 672–675.
[15] ——, "Rival penalized competitive learning for clustering analysis, RBF net, and curve detection," *IEEE Trans. Neural Netw.*, vol. 4, no. 4, pp. 636–648, Jul. 1993.
[16] J. Mao and A. K. Jain, "A self-organizing network for hyperellipsoidal clustering (HEC)," *IEEE Trans. Neural Netw.*, vol. 7, no. 1, pp. 16–29, Jan. 1996.
[17] A. Likas, "A reinforcement learning approach to online clustering," *Neural Comput.*, vol. 11, no. 8, pp. 1915–1932, Nov. 1999.
[18] R. Li, E. Sherrod, J. Kim, and G. Pan, "Fast image vector quantization using a modified competitive learning neural network approach," *Int. J. Imaging Syst. Technol.*, vol. 8, no. 4, pp. 413–418, 1997.
[19] S. A. Billings and G. L. Zheng, "Radial basis function network configuration using genetic algorithms," *Neural Netw.*, vol. 8, no. 6, pp. 877–890, 1995.
[20] A. G. Bors and I. Pitas, "Median radial basis function neural network," *IEEE Trans. Neural Netw.*, vol. 7, no. 6, pp. 1351–1364, Nov. 1996.
[21] A. Krzyzak, T. Linder, and G. Lugosi, "Nonparametric estimation and classification using radial basis function nets and empirical risk minimization," *IEEE Trans. Neural Netw.*, vol. 7, no. 2, pp. 475–487, Mar. 1996.
[22] P. R. Chang and W. H. Yang, "Environment-adaptation mobile radio propagation prediction using radial basis function neural networks," *IEEE Trans. Veh. Technol.*, vol. 46, no. 1, pp. 155–160, Feb. 1997.
[23] A. Roy, S. Govil, and R. Miranda, "A neural-network learning theory and a polynomial time RBF algorithm," *IEEE Trans. Neural Netw.*, vol. 8, no. 6, pp. 1301–1313, Nov. 1997.
[24] J. Lee, C. Beach, and N. Tepedelenlioglu, "A practical radial basis function equalizer," *IEEE Trans. Neural Netw.*, vol. 10, no. 2, pp. 450–455, Mar. 1999.
[25] A. G. Bors and I. Pitas, "Optical flow estimation and moving object segmentation based on median radial basis function network," *IEEE Trans. Image Process.*, vol. 7, no. 5, pp. 693–702, May 1998.
[26] H. Furukawa, T. Ueda, and M. Kitamura, "A systematic method for rational definition of plant diagnostic symptoms by self-organizing neural networks," *Neurocomputing*, vol. 13, no. 2–4, pp. 171–183, Oct. 1996.
[27] Y. M. Cheung, W. M. Leung, and L. Xu, "Adaptive rival penalized competitive learning and combined linear predictor model for financial forecast and investment," *Int. J. Neural Syst.*, vol. 18, no. 5/6, pp. 517–534, 1997.
[28] Y. M. Cheung and L. Xu, "A RPCL-based approach for Markov model identification with unknown state number," *IEEE Signal Process. Lett.*, vol. 7, no. 10, pp. 284–287, Oct. 2000.
[29] Y. M. Cheung, "Rival penalization controlled competitive learning for data clustering with unknown cluster number," in *Proc. 9th ICONIP*, Singapore, Nov. 18–22, 2002, vol. 2, pp. 467–471.
[30] Z. Y. Liu, K. C. Chiu, and L. Xu, "Strip line detection and thinning by RPCL-based local PCA," *Pattern Recognit. Lett.*, vol. 24, no. 14, pp. 2335–2344, Oct. 2003.
[31] L. Xu, "YING-YANG machine: A Bayesian–Kullback scheme for unified learning and new results on vector quantization," in *Proc. ICONIP*, Beijing, China, Oct. 30–Nov. 3, 1995, vol. 2, pp. 977–988. Keynote talk.
[32] ——, "Bayesian–Kullback Ying-Yang machine: Reviews and new results," in *Proc. ICONIP*, Hong Kong, Sep. 24–27, 1996, vol. 1, pp. 59–67.
[33] ——, "Rival penalized competitive learning, finite mixture, and multisets clustering," in *Proc. IEEE Int. Joint Conf. Neural Networks*, Anchorage, AK, May 4–9, 1998, vol. 3, pp. 251–2530.
[34] ——, "An overview on unsupervised learning from data mining perspective," in *Advances in Self-Organizing Maps*. N. Allison *et al.*, Eds. New York: Springer-Verlag, Jun. 2001, pp. 181–210.
[35] ——, "Best harmony, unified RPCL and automated model selection for unsupervised and supervised learning on Gaussian mixtures, three-layer nets and ME-RBF-SVM models," *Int. J. Neural Syst.*, vol. 11, no. 1, pp. 3–69, 2001.
[36] ——, "BYY harmony learning, structural RPCL, and topological self-organizing on mixture models," *Neural Netw.*, vol. 15, no. 8/9, pp. 1125–1151, Oct./Nov. 2002.
[37] D. M. Clark and K. Ravishankar, "A convergence theorem for Grossberg learning," *Neural Netw.*, vol. 3, no. 1, pp. 87–92, 1990.
[38] C. M. Kuan and K. Hornik, "Convergence of learning algorithms with constant learning rates," *IEEE Trans. Neural Netw.*, vol. 2, no. 5, pp. 484–489, Sep. 1991.
[39] B. Kosko, "Stochastic competitive learning," *IEEE Trans. Neural Netw.*, vol. 2, no. 5, pp. 522–529, Sep. 1991.
[40] H. Yang and T. S. Dillon, "Convergence of self-organizing algorithms," *Neural Netw.*, vol. 5, no. 3, pp. 485–493, 1992.
[41] A. S. Galanopoulos, R. L. Moses, and S. C. Ahalt, "Diffusion approximation of frequency sensitive competitive learning," *IEEE Trans. Neural Netw.*, vol. 8, no. 5, pp. 1026–1030, Sep. 1997.
[42] Y. M. Cheung, "Maximum weighted likelihood via rival penalized EM for density mixture clustering with automated model selection," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 750–761, Jun. 2005.
[43] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Stat.*, vol. 22, no. 3, pp. 400–407, Sep. 1951.
[44] S. Y. Kung, *Digital Neural Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
[45] V. A. Zorich, *Mathematical Analysis I*. Berlin, Germany: Springer-Verlag, 2004.
[46] L. Ljung, "Strong convergence of a stochastic approximation algorithm," *Ann. Stat.*, vol. 6, no. 3, pp. 680–696, 1978.
[47] D. E. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[48] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, Dec. 1997.
[49] N. Boujemaa, "Generalized competitive clustering for image segmentation," in *Proc. 19th Int. Conf. North American Fuzzy Information Processing Society*, Atlanta, GA, 2000, pp. 133–137.
[50] C. L. Blake and C. J. Merz, (1998), *UCI Repository of Machine Learning Databases*, Irvine: Dept. Inf. Comput. Sci., Univ. California. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html
[51] S. J. Roberts, R. Everson, and I. Rezek, "Maximum certainty data partitioning," *Pattern Recognit.*, vol. 33, no. 5, pp. 833–839, May 2000.

Jinwen Ma received the M.S. degree in applied mathematics from Xi'an Jiaotong University, Xian, China, in 1988 and the Ph.D. degree in probability theory and statistics from Nankai University, Tianjin China, in 1992.

From July 1992 to November 1999, he was a Lecturer or Associate Professor at the Department of Mathematics, Shantou University. From December 1999, he worked as a Full Professor at the Institute of Mathematics, Shantou University. Since September 2001, he has been with the Department of Information Science, School of Mathematical Sciences, Peking University. During 1995 and 2003, he also worked at or visited several times the Department of Computer Science and Engineering, the Chinese University of Hong Kong, as a Research Associate or Fellow. From September 2005 to August 2006, he was a Research Scientist at the Laboratory of Mathematical Neuroscience, RIKEN Brain Science Institute, Japan. He is the author or coauthor of more than 80 academic papers on neural networks, pattern recognition, artificial intelligence, and information theory.



Taijun Wang received the M.S. degree in signal and information processing from Southeast University, Nanjing, China, in 1982.

Since 1988, he has been an Associate Professor in the Department of Radio Engineering, Southeast University. His research interests are related to digital signal processing, pattern recognition, artificial neural networks, statistical learning theory, and scientific visualization.