On Rival Penalization Controlled Competitive Learning for Clustering with Automatic Cluster Number Selection

Yiu-ming Cheung

Abstract—The existing *Rival Penalized Competitive Learning* (RPCL) algorithm and its variants have provided an attractive way to perform data clustering without knowing the exact number of clusters. However, their performance is sensitive to the preselection of the rival delearning rate. In this paper, we further investigate the RPCL and present a mechanism to control the strength of rival penalization dynamically. Consequently, we propose the Rival Penalization Controlled Competitive Learning (RPCCL) algorithm and its stochastic version. In each of these algorithms, the selection of the delearning rate is circumvented using a novel technique. We compare the performance of RPCCL to RPCL in Gaussian mixture clustering and color image segmentation, respectively. The experiments have produced the promising results.

Index Terms—Rival Penalization Controlled Competitive Learning, stochastic RPCL, clustering, cluster number.

1 Introduction

Competitive learning has been widely applied to a variety of applications such as vector quantization [9], [14], data visualization [8], [13], and particularly to unsupervised clustering [1], [6], [21], [24]. In the literature, k-means [15] is a popular competitive learning algorithm, which trains k seed points (also called *units* hereinafter), denoted as $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_k$, in a way that they converge to the data cluster centers by minimizing the mean-square-error (MSE) function. In general, k-means algorithm has at least two major drawbacks: 1) It suffers from the dead-unit problem. If the initial positions of some units are far away from the inputs (also called *data points* interchangeably) in Euclidean space compared to the other units, these distant units will have no opportunity to be trained and, therefore, immediately become dead units. 2) If the number of clusters is misspecified, i.e., k is not equal to the true cluster number k^*, the performance of k-means algorithm deteriorates rapidly. Eventually, some of the seed points are not located at the centers of the corresponding clusters. Instead, they are either at some boundary points between different clusters or at points biased from some cluster centers [24].

To solve the dead-unit problem, the heuristic Frequency Sensitive Competitive Learning (FSCL) algorithm [1] is an efficient method that circumvents the dead units by adding a relative winning frequency term into the similarity measurement between an input and the seed points. The larger the winning frequency of a seed point, the more it is penalized. Eventually, all units have the opportunity to be updated in the training process. Nevertheless, the FSCL suffers from the same second problem as k-means. That is, the performance of FSCL deteriorates rapidly if k is not well-specified. In the past, some methods have been proposed for cluster number selection. For instance, there have been some statistics, e.g., AIC [2], [3], CAIC [4] and SIC [18], proposed for model selection by formulating the cluster number selection as the choice of component number in a finite mixture model. However, these statistics may overestimate or underestimate the cluster number due to the difficulty of choosing an appropriate penalty function. Another example is the Probabilistic Validation (PV) approach [11] that performs clustering analysis by projecting the high-dimension inputs into one dimension via maximizing the projection indices. It has been shown that the PV can determine the correct number of clusters with a high probability. However, this algorithm is only applicable to linear-separable problems with few clusters. Otherwise, its two-level clustering validation procedure will become rather time-consuming, and the probability of finding the correct number of clusters decreases. Recently, Lange et al. [12] has proposed a Stability Measure to estimate the correct number of clusters. The empirical studies in [12] have shown that this measure outperforms the existing Gap Statistic [19], Prediction Strength [5], [20] and Clest [7]. Also, Xu [22], [23] has proposed a statistic from the Ying-Yang Machine to select the number of clusters in a large sample set. Later, Guo et al. [10] further studied this statistic in a small set of samples. In general, the selection procedure of this statistic is as laborious as the Stability Measure because they both have to perform clustering repeatedly for each possible value of k in a preassigned range.

In contrast, an alternative promising way is to develop some robust algorithms that perform clustering without knowing the exact cluster number. In the past, the typical incremental clustering gradually increases the number k of clusters under the control of a threshold value, which is unfortunately hard to be decided. Recently, Olman et al. [17] has presented a clustering algorithm by formulating a cluster identification problem as searching for substrings with special properties in a linear sequence. They have developed a method for assessing the statistical significance of each identified cluster in a noisy background, through which some accidental data clusters can be ruled out, i.e., the number of clusters can be determined. The performance of the proposed algorithm has been successfully demonstrated in binding site identification. Nevertheless, it is still desired to further investigate this heuristic cluster assessing method on synthetic and real-life data. In the literature, another attractive clustering algorithm is the RPCL [24] and its variants. The basic idea in each of them is that, for each input, not only the winner of the seed points is updated to adapt to the input, but also its nearest rival (i.e., the second winner) is delearned by a smaller learning rate (also called *delearning rate* hereinafter). Empirical studies have shown that the RPCL can automatically select the correct cluster number by gradually driving redundant seed points far away from the input dense regions. However, some experiments have also found that the performance of RPCL is sensitive to the delearning rate. In general, the RPCL will fail to perform correct clustering if the delearning rate is not well preassigned. To our best knowledge, it is definitely a nontrivial task to select this rate in advance.

In this paper, we will concentrate on studying the RPCL algorithm and propose a novel technique to circumvent the selection of the delearning rate. We have noticed that the RPCL [24] adaptively performs the rival penalization for each input without considering the distance from the winning unit to the rival. In fact, the rival should be more penalized if its distance to the winner is closer than the one between the winner and the input. This idea is analogous to the social scenario in our daily life. For example, the competition between two candidates called A and B (we assume that A is the final *winner* and B is therefore *the rival*) in an election campaign will become more intense if their public opinion polls are closer. Otherwise, A will be almost sure to win the election with little attack against (i.e., little penalizing) B during the election campaign. Based on this idea, we therefore present a mechanism, in which the rival-penalized strength is dynamically adjusted based on the relative distance of the winner to the rival and the current input, respectively. We have associated this mechanism with the delearning rule of the RPCL, through which a new improved algorithm named *Rival Penalization Controlled Competitive Learning* (RPCCL) is proposed. This new algorithm always sets the delearning rate at the same value as the learning rate. Hence, the selection of the delearning rate is novelly

• *The author is with the Department of Computer Science, Hong Kong Baptist University, Rm. 709, 7/F, Sir Run Run Shaw Building, Kowloon, Hong Kong, P.R. China. E-mail: ymc@comp.hkbu.edu.hk.*

Manuscript received 18 June 2004; revised 14 Mar. 2005; accepted 7 July 2005; published online 19 Sept. 2005.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0177-0604.

circumvented. In contrast, such a setting will always make the RPCL fail as pointed out in [24]. In the RPCCL, a rival seed point is always penalized with the delearning rule. Actually, we can also perform the rival penalization in a probabilistic way. Subsequently, we obtain a variant named *Stochastic RPCL* (S-RPCL) algorithm, which penalizes the rivals by using the same rule as the RPCL, but the penalization is performed stochastically. In effect, this learning procedure is equivalent to the one in the RPCCL. Hence, both of the RPCCL and S-RPCL will finally lead to the same clustering result. We compare the performance of RPCCL to the RPCL in Gaussian mixture clustering and color image segmentation, respectively. The experiments have shown that the RPCCL can smoothly work in all cases we have tried so far, but the RPCL may not. Furthermore, we find that the RPCCL often gives correct clustering results much faster than the RPCL because the strength of rival penalization in the former is stronger on average.

The remainder of this paper is organized as follows: Section 2 overviews the RPCL algorithm in data clustering. Section 3 elaborates the RPCCL in detail, including the embedded rival-penalization controlling mechanism and its stochastic implementation. We empirically show the performance of RPCCL in comparison with the RPCL in Section 4. Finally, we draw a conclusion in Section 5.

2 Overview of RPCL Algorithm in Data Clustering

Suppose N inputs, $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N}$, come from k^* unknown clusters. The RPCL randomly initializes k seed points: $\mathbf{m_1}, \mathbf{m_2}, \ldots, \mathbf{m_k}$, and adaptively updates them so that those $\mathbf{x_t}$s can be correctly classified based on the indicator function

$$g(j|\mathbf{x}_t) = \begin{cases} 1, & \text{if } j = c = \arg\min_{1\le i\le k} \|\mathbf{x}_t - \mathbf{m}_i\|^2 \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

That is, $\mathbf{x}_t$ is classified into the jth cluster if $g(j|\mathbf{x}_t) = 1$. The basic idea of RPCL [24] to update the seed points is that for each input, not only the winning seed point $\mathbf{m}_c$ (i.e., $g(c|\mathbf{x}_t) = 1$) is modified to adapt to the input, but also its the nearest rival is delearned by a smaller learning rate. Specifically, the algorithm is:

Step A.1: Randomly take a sample $\mathbf{x}_t$ from the data set $D = \{\mathbf{x}_t\}_{t=1}^N$, and for $j = 1, 2, \ldots, k$, let

$$I(j|\mathbf{x}_t) = \begin{cases} 1, & \text{if } j = c \text{ with } c = \arg\min_i \gamma_i \{\|\mathbf{x}_t - \mathbf{m}_i\|^2 \\ -1, & \text{if } j = r \text{ with } r = \arg\min_{i\ne c} \gamma_i \|\mathbf{x}_t - \mathbf{m}_i\|^2 \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

with

$$\gamma_i = \frac{n_i}{\sum_{u=1}^k n_u}, \quad (3)$$

where n_i is the cumulative number of the winning occurrences of $\mathbf{m}_i$ in the past.

Step A.2: Update $\mathrm{n_c}$, the winner $\mathbf{m}_c$, and its nearest rival $\mathbf{m}_r$ by

$$\begin{aligned} \mathrm{n_c^{new}} &= \mathrm{n_c^{old}} + 1 \\ \mathbf{m}_\tau^{\text{new}} &= \mathbf{m}_\tau^{\text{old}} + \Delta\mathbf{m}_\tau, \quad \tau = c, r \end{aligned} \quad (4)$$

with

$$\begin{aligned} \Delta\mathbf{m}_c &= \alpha_c(\mathbf{x}_t - \mathbf{m}_c) \\ \Delta\mathbf{m}_r &= -\alpha_r(\mathbf{x}_t - \mathbf{m}_r), \end{aligned} \quad (5)$$

while the other seed points are unchanged. In (5), α_c is a small positive learning rate, whereas α_r is called the *delearning rate*, whose value is generally much smaller than α_c. The above two steps iterate for each input until a stop criterion is satisfied, e.g., the iteration number reaches the preassigned maximum value, or absolute difference between the consecutive classification errors is smaller than a preassigned threshold value.

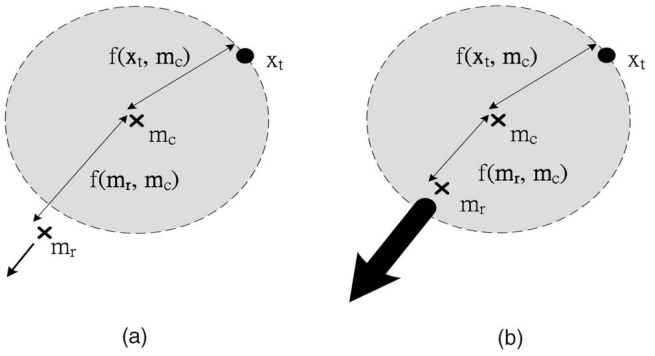


Fig. 1. The rival penalization controlling mechanism in the RPCCL algorithm. It can be seen that: (a) When $f(\mathbf{m}_r, \mathbf{m}_c) > f(\mathbf{x}_t, \mathbf{m}_c)$, the rival penalization gradually decreases as the distance between the rival and the winner increases. (b) When $f(\mathbf{m}_r, \mathbf{m}_c) \le f(\mathbf{x}_t, \mathbf{m}_c)$, a full penalization is applied to the rival seed point $\mathbf{m}_r$, i.e., the rival-penalized strength reaches its maximum value α_c.

As shown in [24], the RPCL can automatically determine the number of clusters by driving extra seed points far away from the input dense regions as long as $k \ge k^*$. However, some empirical studies have also found that the performance of RPCL is sensitive to the selection of the delearning rate α_r. If α_r is too small, the RPCL may not have enough force to push the redundant seed points away from the input regions. On the other hand, if α_r is too large, the RPCL will push almost all seed points far away from the inputs because of too strong penalization strength. Roughly speaking, an appropriate value of α_r not only varies with the different clustering problems, but is also related to the initial positions of the seed points. Such a selection is definitely a nontrivial task. In the past, α_r is mostly selected by trial and error, which thereby limits the great potential of RPCL in real applications.

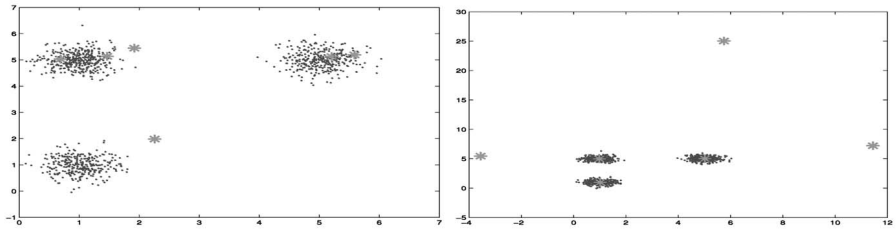
In the following, we will present the RPCCL that utilizes a mechanism to control the rival-penalized strength. Consequently, it can circumvent the selection of α_r by always setting α_r at the same value as α_c.

3 RPCCL Algorithm and its Stochastic Implementation

The RPCCL algorithm utilizes a novel mechanism to control the rival penalization. The idea of this mechanism is that the rival should be fully penalized if the winner suffers from the severe competition from the rival; otherwise, the penalization strength should be proportional to the degree of competition level. To realize this idea, given an input $\mathbf{x}_t$, we define the competition to be severe if the distance of the winner to the rival is closer than its distance to $\mathbf{x}_t$. Furthermore, we define the full-penalization strength (i.e., the maximum rival-penalized strength) of the rival to be α_c upon the fact that α_r is generally smaller than α_c in the RPCL. Subsequently, we give out this penalization control mechanism by $\alpha_c p(\mathbf{x}_t; \mathbf{m}_c, \mathbf{m}_r)$ with

$$p(\mathbf{x}_t; \mathbf{m}_c, \mathbf{m}_r) = \frac{\min[f(\mathbf{m}_r, \mathbf{m}_c), f(\mathbf{x}_t, \mathbf{m}_c)]}{f(\mathbf{m}_r, \mathbf{m}_c)}, \quad (6)$$

where f is a certain distance measuring function, e.g., Euclidean distance, or more general Mahalanobis distance. Since the numerator of (6) is always smaller than or equal to the denominator, the value of $p(\mathbf{x}_t; \mathbf{m}_c, \mathbf{m}_r)$ must be between 0 and 1. It can be seen that, as illustrated in Fig. 1, the rival will be fully penalized with the rate α_c as $f(\mathbf{m}_r, \mathbf{m}_c) \le f(\mathbf{x}_t, \mathbf{m}_c)$. Otherwise, the rival penalization is gradually attenuated when the distance between the rival and the winner increases. Hereinafter, we will



(a) (b)

Fig. 2. The positions of six seed points marked by "*" in the input space at Experiment 1 of Section 4.1.1: (a) the initial positions, and (b) the final positions obtained via the RPCCL.

measure the distance between two points by using the Euclidean distance only, i.e., (6) can be specified as:

$$p(\mathbf{x}_t;\mathbf{m}_c,\mathbf{m}_r)=\frac{\min(\|\mathbf{m}_r-\mathbf{m}_c\|,\|\mathbf{x}_t-\mathbf{m}_c\|)}{\|\mathbf{m}_r-\mathbf{m}_c\|}. \tag{7}$$

By embedding the control mechanism into the rival's update, we then obtain the RPCCL algorithm as follows:

Step B.1: Randomly take an input $\mathbf{x}_t$ from the data set D, calculate $I(j|\mathbf{x}_t)$ by (2).

Step B.2: Update $\mathrm{n_c}$, the winner $\mathbf{m}_c$ and the rival $\mathbf{m}_r$ only by (4), but (5) becomes

$$\begin{aligned}\Delta\mathbf{m}_c&=\alpha_c(\mathbf{x}_t-\mathbf{m}_c)\\ \Delta\mathbf{m}_r&=-\alpha_c p(\mathbf{x}_t;\mathbf{m}_c,\mathbf{m}_r)(\mathbf{x}_t-\mathbf{m}_r)\\ &=-\alpha_c\frac{\min(\|\mathbf{m}_r-\mathbf{m}_c\|,\|\mathbf{x}_t-\mathbf{m}_c\|)}{\|\mathbf{m}_r-\mathbf{m}_c\|}(\mathbf{x}_t-\mathbf{m}_r).\end{aligned} \tag{8}$$

These two steps iterate for each input until a stop criterion stated previously is satisfied.

From (8), it can be seen that the rival will be fully penalized with the rate α_c as $\|\mathbf{m}_c-\mathbf{m}_r\|\le\|\mathbf{m}_c-\mathbf{x}_t\|$. Otherwise, the rival will be penalized with the strength $\alpha_c\frac{\|\mathbf{m}_c-\mathbf{x}_t\|}{\|\mathbf{m}_c-\mathbf{m}_r\|}$, which is gradually attenuated as $\|\mathbf{m}_c-\mathbf{m}_r\|$ increases. On average, the rival-penalized strength in (8) is no less than $0.25\alpha_c$ if there are two or more seed points located in one cluster in which the data points uniformly distributed. In contrast, such a large rival-penalized strength will make the RPCL break down completely. Hence, α_r in (5) must be much smaller than $0.25\alpha_c$. Consequently, the RPCCL generally drives the redundant seed points far away from the clusters much faster than the RPCL. In fact, if we further fix $p(\mathbf{x}_t;\mathbf{m}_c,\mathbf{m}_r)$ at an appropriate value smaller than 1, (8) is then equal to (5) with $\alpha_r=\alpha_c p(\mathbf{x}_t;\mathbf{m}_c,\mathbf{m}_r)$. That is, the RPCCL is actually a generalization of the RPCL.

Furthermore, we have noticed that $p(\mathbf{x}_t;\mathbf{m}_c,\mathbf{m}_r)$ in (8) is always between 0 and 1. It can therefore be regarded as the probability of rival penalization. As a result, we can alternatively give out a stochastic version of the RPCCL named Stochastic RPCL (S-RPCL) which, in effect, is identical to the RPCCL. The detailed algorithm is as follows:

Step C.1: Randomly take an input $\mathbf{x}_t$ from the data set D, calculate $I(j|\mathbf{x}_t)$ by (2).

Step C.2: Update $\mathrm{n_c}$ and $\mathbf{m}_c$ by (4) and (8). Then, we generate a uniformly-distributed random number $\nu\in[0,1]$. Let

$$\varrho(\mathbf{x}_t)=\begin{cases}1, & \text{if } \nu\le p(\mathbf{x}_t;\mathbf{m}_c,\mathbf{m}_r);\\ 0, & \text{otherwise.}\end{cases} \tag{9}$$

We update the rival $\mathbf{m}_r$ by

$$\Delta\mathbf{m}_r=-\alpha_c\varrho(\mathbf{x}_t)(\mathbf{x}_t-\mathbf{m}_r). \tag{10}$$

Step C.1 and **Step C.2** iterate for each input until a stop criterion is satisfied.

In (9), the value of $\varrho(\mathbf{x}_t)$ is switched between 0 and 1, which causes the rival penalization in (10) to be implemented discontinuously. Actually, it can be seen that (10) is exactly equal to the delearning rule of the RPCL in (5) with the delearning rate $\alpha_r=\alpha_c$ if $\varrho=1$. As pointed out in [24], we have known that the RPCL will completely break down if $\alpha_r=\alpha_c$. But, one interesting thing is that the S-RPCL can work very well as long as the rival penalization can be discontinuously implemented in a controlled way. As the S-RPCL has the same clustering performance as the RPCCL, we will demonstrate the performance of RPCCL in comparison with the RPCL only in the next section.

4 Experimental Results

We investigated the performance of RPCCL on Gaussian mixture clustering and color image segmentation in comparison with the RPCL. In all the experiments, we set the learning rate α_c at 0.001, and α_r at 0.0001 by default in implementing the RPCL.

4.1 RPCCL and RPCL in Gaussian Mixture Clustering

4.1.1 Experiment 1

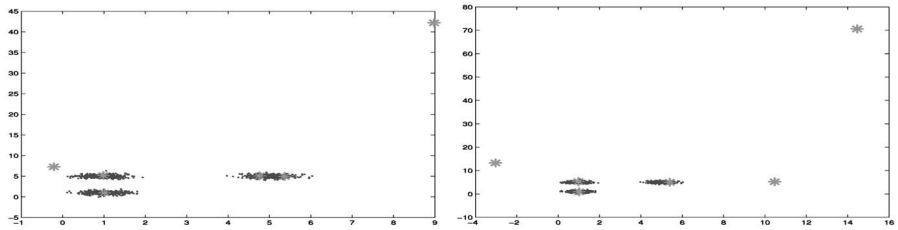
We used the 1,000 synthetic data points from a mixture of three Gaussian densities:

$$\begin{aligned}p(\mathbf{x})=&\,0.3G\left[\mathbf{x}|\begin{pmatrix}1\\1\end{pmatrix},\begin{pmatrix}0.1, & 0\\ 0, & 0.1\end{pmatrix}\right]\\ &+0.4G\left[\mathbf{x}|\begin{pmatrix}1\\5\end{pmatrix},\begin{pmatrix}0.1, & 0\\ 0, & 0.1\end{pmatrix}\right]\\ &+0.3G\left[\mathbf{x}|\begin{pmatrix}5\\5\end{pmatrix},\begin{pmatrix}0.1, & 0\\ 0, & 0.1\end{pmatrix}\right],\end{aligned} \tag{11}$$

where $G(\mathbf{x}|\mathbf{m},\Sigma)$ denotes the probability density function of the variable $\mathbf{x}$ with the mean $\mathbf{m}$ and the covariance matrix Σ. As shown in Fig. 2a, the data form three well-separated clusters. We randomly located six seed points $\mathbf{m}_1,\mathbf{m}_2,\ldots,\mathbf{m}_6$ at:

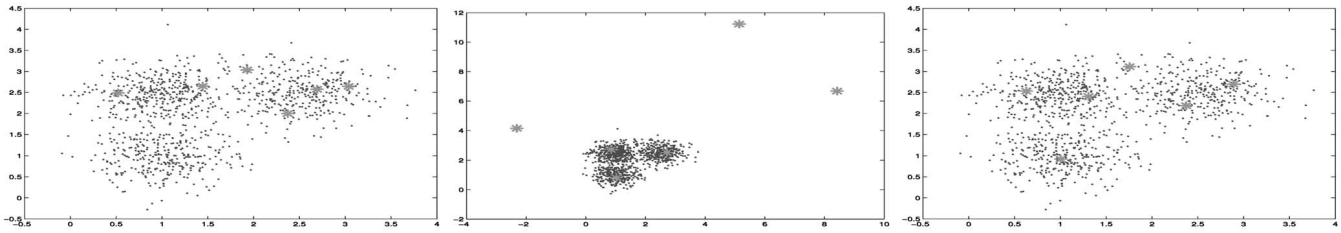
$$\begin{aligned}&\mathbf{m}_1=\begin{pmatrix}2.2580\\1.9849\end{pmatrix},\quad \mathbf{m}_2=\begin{pmatrix}1.4659\\5.1359\end{pmatrix},\quad \mathbf{m}_3=\begin{pmatrix}0.6893\\5.0331\end{pmatrix}\\ &\mathbf{m}_4=\begin{pmatrix}5.2045\\5.1298\end{pmatrix},\quad \mathbf{m}_5=\begin{pmatrix}1.9193\\5.4489\end{pmatrix},\quad \mathbf{m}_6=\begin{pmatrix}5.5869\\5.1937\end{pmatrix}.\end{aligned} \tag{12}$$

After 100 epochs, i.e., repeatedly scanning all available data points 100 times, the six seed points learned by the RPCCL had been converged to:



(a) (b)

Fig. 3. The experimental results of the RPCL in Experiment 1 of Section 4.1.1. (a) The final positions of six seed points obtained via the RPCL with 100 epochs, where only two seed points are pushed far away from the input dense regions; (b) The final positions obtained via the RPCL with 200 epochs, where all extra seed points have been successfully driven away from the input regions.



(a) (b) (c)

Fig. 4. The positions of six seed points marked by "*" in the input space at Experiment 2 of Section 4.1.1: (a) The initial positions, (b) the final positions obtained via the RPCCL, and (c) the final positions obtained via the RPCL.

$$\mathbf{m}_1 = \begin{pmatrix} 1.0131 \\ 0.9806 \end{pmatrix}, \quad \mathbf{m}_2 = \begin{pmatrix} 0.9845 \\ 4.9823 \end{pmatrix}, \quad \mathbf{m}_3 = \begin{pmatrix} -3.5557 \\ 5.4466 \end{pmatrix}$$
$$\mathbf{m}_4 = \begin{pmatrix} 5.0180 \\ 5.0043 \end{pmatrix}, \quad \mathbf{m}_5 = \begin{pmatrix} 5.7498 \\ 25.0371 \end{pmatrix}, \quad \mathbf{m}_6 = \begin{pmatrix} 11.4483 \\ 7.2208 \end{pmatrix}. \tag{13}$$

As shown in Fig. 2b, the RPCCL has successfully put three seed points: $\mathbf{m}_1$, $\mathbf{m}_2$, and $\mathbf{m}_4$ into the appropriate positions of three clusters, meanwhile driving the other three extra seed points: $\mathbf{m}_3$, $\mathbf{m}_5$, and $\mathbf{m}_6$ far away from the input dense regions. In contrast, after 100 epochs, the RPCL just led the six seed points to:

$$\mathbf{m}_1 = \begin{pmatrix} 1.0131 \\ 0.9806 \end{pmatrix}, \quad \mathbf{m}_2 = \begin{pmatrix} 0.9862 \\ 5.1899 \end{pmatrix}, \quad \mathbf{m}_3 = \begin{pmatrix} -0.2110 \\ 7.2922 \end{pmatrix}$$
$$\mathbf{m}_4 = \begin{pmatrix} 4.7637 \\ 5.0947 \end{pmatrix}, \quad \mathbf{m}_5 = \begin{pmatrix} 8.9795 \\ 42.2057 \end{pmatrix}, \quad \mathbf{m}_6 = \begin{pmatrix} 5.3665 \\ 4.9695 \end{pmatrix}. \tag{14}$$

As shown in Fig. 3a, the RPCL pushed only two seed points $\mathbf{m}_3$ and $\mathbf{m}_5$ far away from the input regions. We then further learned the seed points up to 200 epochs. It was found that the RPCL gradually drove the three extra points $\mathbf{m}_3$, $\mathbf{m}_5$, and $\mathbf{m}_6$ to

$$\mathbf{m}_3 = \begin{pmatrix} -3.0326 \\ 13.2891 \end{pmatrix} \quad \mathbf{m}_5 = \begin{pmatrix} 14.4600 \\ 70.6014 \end{pmatrix}, \quad \mathbf{m}_6 = \begin{pmatrix} 10.4714 \\ 5.2240 \end{pmatrix}, \tag{15}$$

which were far away from the input dense regions as shown in Fig. 3b, while the other three seed points located at the correct positions as follows:

$$\mathbf{m}_1 = \begin{pmatrix} 1.0167 \\ 0.9321 \end{pmatrix} \quad \mathbf{m}_2 = \begin{pmatrix} 0.9752 \\ 5.3068 \end{pmatrix}, \quad \mathbf{m}_4 = \begin{pmatrix} 5.4022 \\ 5.0054 \end{pmatrix}. \tag{16}$$

It can be seen that the RPCL can finally work well in this case, but it needs more computing costs in comparison with the RPCCL.

4.1.2 Experiment 2

We further investigated the RPCCL and RPCL on the overlapping data clusters as shown in Fig. 4a. After 100 epochs, we found that the RPCCL had given out the correct results as shown in Fig. 4b, but the RPCL could not work as shown in Fig. 4c even if we increased the epoch number to $1,000$. Under the circumstances, we further investigated the RPCL by adjusting α_r along two directions: from 0.0001 to 0.0009, and from 0.0001 to 0.00001, respectively, with a constant step: 0.00001. Unfortunately, we could not find out an appropriate α_r in all cases we had tried so far to make the RPCL work.

4.2 RPCCL and RPCL in Color Image Segmentation

Image segmentation is the process of segmenting an image into different homogeneous regions, which is a critical step in image analysis and pattern recognition. In the literature, a variety of approaches have been proposed to deal with gray-scale images such as edge-based, region-based, and pixel-based approaches. Some of them can also be used in color images. In this paper, we focus on pixel-based segmentation only for color image segmentation. We operate in the simple Red-Green-Blue (RGB) color space model that represents each pixel in an image by the three-color components. Assuming homogenous objects have similar colors, we can therefore group pixels of similar colors into the same cluster based on a certain distance measure over the three-dimensional RGB color space. Eventually, given a set of image pixels, denoted as $D = \{\mathbf{x}_t\}_{t=1}^{N}$, where $\mathbf{x}_t = (x_t^R, x_t^G, x_t^B)^T$ is a 3×1 vector representing a color pixel in RGB space, color image segmentation based on pixels can be then formulated as a three-dimensional data clustering problem. In the following, we will perform color image segmentation by using the RPCCL and RPCL, respectively.

4.2.1 Experiment 1

We used the *House* image with 128×128 pixels to compare the segmentation performance of the RPCCL and RPCL. The original House image is shown in Fig. 5a. We randomly initialized 30 seed points in the RGB color space. After the algorithm performance

(a) (b) (c)

Fig. 5. Segmentation results of the House image: (a) The original House image, (b) the results by the RPCL with 30 seed points, where the texture of the red wall is still retained, and (c) the results by the RPCCL with 18 seed points only, where the wall texture is successfully removed.

(a) (b) (c)

Fig. 6. Segmentation results of the Audience image: (a) The original Audience image, (b) the results from the RPCL with 29 seed points, and (c) the results from the RPCCL with 23 seed points only.

converged, we found that the RPCL failed to drive out any seed point from the input regions. This phenomenon shows again that the RPCL performance is sensitive to the selection of the delearning rate. In contrast, the RPCCL had driven out 12 extra seed points far away from the input regions. This implies that the RPCCL utilizes a smaller set of seed points in the image segmentation. Figs. 5b and 5c show a snapshot of the segmentation results at Epoch 100 via these two algorithms, respectively. It can be seen that the RPCL still retains the texture of the red wall, but the RPCCL has successfully removed it. That is, the results from the RPCCL are better than the RPCL.

4.2.2 Experiment 2

Similar to Experiment 1 of Section 4.2.1, we let the number of seed points be 30, and used another *Audience* image with 128×128 pixels to compare their segmentation performance. The original Audience image is shown in Fig. 6a. After the algorithm performance converged, we found that the RPCL had driven out one seed point from the input regions, whereas the RPCCL had driven out seven extra seed points far away from the data points. Once again, the RPCCL utilized a smaller set of seed points in this trial. A snapshot of the segmentation results via these two algorithms at Epoch 50 is shown in Figs. 6b and 6c. It can be seen that the sparse points in the audience hindbrain are still retained in the result of RPCCL, but missed in the one of RPCL. This implies that the RPCCL performance was slightly better than the RPCL, although both of them led to the similar results.

Further, when adjusting the delearning rate of RPCL up to 0.0002, we found that the RPCL could finally drive out six extra seed points away from the data points. This scenario shows again that the performance of RPCL is sensitive to the value of the delearning rate.

5 Conclusion

We have further investigated the RPCL with presenting a mechanism to dynamically control the rival-penalized strength. Consequently, we have proposed an improved algorithm named rival penalized controlled competitive learning (RPCCL) and its equivalent stochastic version. Both of them have novelly circumvented the difficult selection of the delearning rate. We have compared the performance of RPCCL with the RPCL in Gaussian mixture clustering and color image segmentation. The numerical simulations have produced the promising results.

Acknowledgments

The author would like to sincerely thank the editor and three anonymous reviewers for their valuable comments and insightful suggestions. Also, thanks go to Mr. Lap-tak Law for helping to carry out the experiments on color image segmentation. This work was supported by a grant from the Research Grant Council of the Hong Kong SAR (Project No: HKBU 2156/04E), and by a Faculty Research Grant of Hong Kong Baptist University (Project Code: FRG/02-03/II-40).

References

[1] S.C. Ahalt, A.K. Krishnamurty, P. Chen, and D.E. Melton, "Competitive Learning Algorithms for Vector Quantization," *Neural Networks,* vol. 3, pp. 277-291, 1990.

[2] H. Akaike, "Information Theory and an Extension of the Maximum Likelihood Principle," *Proc. Second Int'l Symp. Information Theory,* pp. 267-281, 1973.

[3] H. Akaike, "A New Look at the Statistical Model Identfication," *IEEE Trans. Automatic Control,* pp. 716-723, 1974.

[4] H. Bozdogan, "Model Selection and Akaike's Information Criterion: The General Theory and Its Analytical Extensions," *Psychometrika,* vol. 52, no. 3, pp. 345-370, 1987.

[5] J. Breckenridge, "Replicating Clustering Analysis: Method, Consistency and Validity," *Multivariate Behavioural Research,* 1989.

[6] E.W. Forgy, "Cluster Analysis of Multivariate Data: Efficiency Versus Interpretability of Classifications," *Proc. Biometric Soc. Meetings*, 1965.
[7] J. Fridlyand and S. Dudoit, "Applications of Resampling Methods to Estimate the Number of Clusters and to Improve the Accuracy of a Clustering Method," Technical Report 600, Statistics Dept., UC Berkeley, Sept. 2001.
[8] B. Fritzke, "Growing Cell Structures—A Self-Organizing Network for Unsupervised and Supervised Learning," *Neural Networks*, vol. 7, no. 9, pp. 1441-1460, 1994.
[9] R.M. Gray, "Vector Quantization," *IEEE ASSP Magazine*, vol. 1, pp. 4-29, 1984.
[10] P. Guo, C.L. Philip Chen, and M.R. Lye, "Cluster Number Selection for a Small Set of Samples Using the Bayesian Ying-Yang Model," *IEEE Trans. Neural Networks*, vol. 13, no. 3, pp. 757-763, 2002.
[11] M. Har-even and V.L. Brailovsky, "Probabilistic Validation Approach for Clustering," *Pattern Recognition Letters*, vol. 16, pp. 1189-1196, 1995.
[12] T. Lange, M.L. Braun, V. Roth, and J.M. Buhmann, "Stability-Based Model Selection," *Proc. Neural Information Processing Systems*, Paper ID: AA17, 2002.
[13] T. Kohonen, "Self-Organized Formation of Topologically Correct Feature Maps," *Biological Cybernetics*, vol. 43, pp. 59-69, 1982.
[14] Y. Linde, A. Buzo, and R.M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. Comm.*, pp. 84-95, 1980.
[15] J.B. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," *Proc. Fifth Berkeley Symp. Math. Statistics and Probability*, vol. 1, Berkeley, Calif.: Univ. of California Press, pp. 281-297, 1967.
[16] T.M. Martinetz and K.J. Schulten, "A 'Neural-Gas' Network Learns Topologies," *Artificial Neural Networks*, pp. 397-402, 1991.
[17] V. Olman, D. Xu, and Y. Xu, "Cubic: Identification of Regulatory Binding Sites Through Data Clustering," *J. Bioinformatics and Computational Biology*, vol. 1, no. 1, pp. 21-40, 2003.
[18] G. Schwarz, "Estimating the Dimension of a Model," *The Annals of Statistics*, vol. 6, no. 2, pp. 461-464, 1978.
[19] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the Number of Clusters via the Gap Statistic," *J. Royal Statistical Soc., Series B*, vol. 63, pp. 411-423, 2001.
[20] R. Tibshirani, G. Walther, D. Botstein, and P. Brown, "Cluster Validation by Prediction Strength," technical report, Statistics Dept., Stanford Univ., Sept., 2001.
[21] H.Q. Wang and D.S. Huang, "A Novel Clustering Analysis Based on PCA and SOMs for Gene Expression Patterns," *Lecture Notes in Computer Science*, vol. 3174, pp. 476-481, Springer-Verlag, 2004.
[22] L. Xu, "How Many Clusters?: A Ying-Yang Machine Based Theory for a Classical Open Problem in Pattern Recognition," *Proc. IEEE Int'l Conf. Neural Networks*, vol. 3, pp. 1546-1551, 1996.
[23] L. Xu, "Bayesian Ying-Yang Machine, Clustering and Number of Clusters," *Pattern Recognition Letters*, vol. 18, nos. 11-13, pp. 1167-1178, 1997.
[24] L. Xu, A. Krzyżak, and E. Oja, "Rival Penalized Competitive Learning for Clustering Analysis, RBF Net, and Curve Detection," *IEEE Trans. Neural Networks*, vol. 4, pp. 636-648, 1993.