LETTER Communicated by Andrew Barto

A Reinforcement Learning Approach to Online Clustering

Aristidis Likas
Department of Computer Science, University of Ioannina, 45110, Ioannina, Greece

A general technique is proposed for embedding online clustering algorithms based on competitive learning in a reinforcement learning framework. The basic idea is that the clustering system can be viewed as a reinforcement learning system that learns through reinforcements to follow the clustering strategy we wish to implement. In this sense, the reinforcement guided competitive learning (RGCL) algorithm is proposed that constitutes a reinforcement-based adaptation of learning vector quantization (LVQ) with enhanced clustering capabilities. In addition, we suggest extensions of RGCL and LVQ that are characterized by the property of sustained exploration and significantly improve the performance of those algorithms, as indicated by experimental tests on well-known data sets.

1 Introduction

Many pattern recognition and data analysis tasks assume no prior class information about the data to be used. Pattern clustering belongs to this category and aims at organizing the data into categories (clusters) so that patterns within a cluster are more similar to each other (in terms of an appropriate distance metric) than patterns belonging to different clusters. To achieve this objective, many clustering strategies are parametric and operate by defining a clustering criterion and then trying to determine the optimal allocation of patterns to clusters with respect to the criterion. In most cases such strategies are iterative and operate online; patterns are considered one at a time, and, based on the distance of the pattern from the cluster centers, the parameters of the clusters are adjusted according to the clustering strategy. In this article, we present an approach to online clustering that treats competitive learning as a reinforcement learning problem. More specifically, we consider partitional clustering (or hard clustering or vector quantization), where the objective is to organize patterns into a small number of clusters such that each pattern belongs exclusively to one cluster.

Reinforcement learning constitutes an intermediate learning paradigm that lies between supervised (with complete class information available) and unsupervised learning (with no available class information). The training information provided to the learning system by the environment (external teacher) is in the form of a scalar reinforcement signal r that constitutes a measure of how well the system operates. The main idea of this article is that the clustering system does not directly implement a prespecified clustering strategy (for example, competitive learning) but instead tries to learn to follow the clustering strategy using the suitably computed reinforcements provided by the environment. In other words, the external environment rewards or penalizes the learning system depending on how well it learns to apply the clustering strategy we have selected to follow. This approach will be formally defined in the following sections and leads to the development of clustering algorithms that exploit the stochasticity inherent in a reinforcement learning system and therefore are more flexible (do not get easily trapped in local minima) compared to the original clustering procedures. The proposed technique can be applied with any online hard clustering strategy and suggests a novel way to implement the strategy (update equations for cluster centers).

In addition we present an extension of the approach that is based on the sustained exploration property that can be easily obtained by a minor modification to the reinforcement update equations and gives the algorithms the ability to escape from local minima.

In the next section we provide a formal definition of online hard clustering as a reinforcement learning problem and present reinforcement learning equations for the update of the cluster centers. The equations are based on the family of REINFORCE algorithms that have been shown to exhibit stochastic hillclimbing properties (Williams, 1992). Section 3 describes the reinforcement guided competitive learning (RGCL) algorithm that constitutes a stochastic version of the learning vector quantization (LVQ) algorithm. Section 4 discusses issues concerning sustained exploration and the adaptation of the reinforcement learning equations to achieve continuous search of the parameter space, section 5 presents experimental results and several comparisons using well-known data sets, and section 6 summarizes the article and provides future research directions.

2 Clustering as a Reinforcement Learning Problem

2.1 Online Competitive Learning. Suppose we are given a sequence $X = (x_1, \ldots, x_N)$ of unlabeled data $x_i = (x_{i1}, \ldots, x_{ip})^\top \in R^p$ and want to assign each of them to one of L clusters. Each cluster i is described by a prototype vector $w_i = (w_{i1}, \ldots, w_{ip})^\top$ $(i = 1, \ldots, L)$, and let $W = (w_1, \ldots, w_L)$. Also let $d(x, w)$ denote the distance metric based on which the clustering is performed. In the case of hard clustering, most methods attempt to find good clusters by minimizing a suitably defined objective function $J(W)$. We restrict ourselves here to techniques based on competitive learning where the objective function is (Kohonen, 1989; Hathaway & Bezdek, 1995)

$$J = \sum_{i=1}^{N} \min_r d(x_i, w_r). \tag{2.1}$$

The clustering strategy of the competitive learning techniques can be summarized as follows:

1. Randomly take a sample x_i from X.
2. Compute the distances $d(x_i, w_j)$ for $j = 1, \ldots, L$ and locate the winning prototype $j^\star$, that is, the one with minimum distance from x_i.
3. Update the weights w_{ij} so that the winning prototype $w_{j^\star}$ moves toward pattern x_i.
4. Go to step 1.

Depending on what happens in step 3 with the nonwinning prototypes, several competitive learning schemes have been proposed such as LVQ (or adaptive k-means) (Kohonen, 1989), the RPCL (rival penalized competitive learning) (Xu, Krzyzak, & Oja, 1993), the SOM network (Kohonen, 1989), the "neural-gas" network (Martinez, Berkovich, & Schulten, 1993), and others. Moreover in step 2, some approaches, such as frequency sensitive competitive learning (FSCL), assume that the winning prototype minimizes a function of the distance $d(x, w)$ and not the distance itself.

2.2 Immediate Reinforcement Learning. In the framework of reinforcement learning, a system accepts inputs from the environment, responds by selecting appropriate actions (decisions), and the environment evaluates the decisions by sending a rewarding or penalizing scalar reinforcement signal. According to the value of the received reinforcement, the learning system updates its parameters so that good decisions become more likely to be made in the future and bad decisions become less likely to occur (Kaelbling, Littman, & Moore, 1996). A simple special case is immediate reinforcement learning, where the reinforcement signal is received at every step immediately after the decision has been made.

In order for the learning system to be able to search for the best decision corresponding to each input, a stochastic exploration mechanism is frequently necessary. For this reason many reinforcement learning algorithms apply to neural networks of stochastic units. These units draw their outputs from some probability distribution, employing either one or many parameters. These parameters depend on the inputs and the network weights and are updated at each step to achieve the learning task. A special case, which is of interest to our approach, is when the output of each unit is discrete and more specifically is either one or zero, depending on a single parameter $p \in [0, 1]$. This type of stochastic unit is called the Bernoulli unit (Barto & Anandan, 1985; Williams, 1988, 1992).

Several training algorithms have been developed for immediate reinforcement problems. We have used the family of REINFORCE algorithms in which the parameters w_{ij} of the stochastic unit i with input x are updated as

$$\Delta w_{ij} = a(r - b_{ij})\frac{\partial \ln g_i}{\partial w_{ij}}, \tag{2.2}$$

where $a > 0$ is the learning rate, r the received reinforcement, and b_{ij} a quantity called the reinforcement baseline. The quantity $\partial \ln g_i/\partial w_{ij}$ is called the characteristic eligibility of w_{ij}, where $g_i(y_i; w_i, x)$ is the probability mass function (in the case of a discrete distribution) or the probability density function (in the case of a continuous distribution), which determines the output y_i of the unit as a function of the parameter vector w_i and the input pattern x to the unit.

An important result is that REINFORCE algorithms are characterized by the stochastic hillclimbing property. At each step, the average update direction $E\{\Delta W \mid W, x\}$ in the weight space lies in the direction for which the performance measure $E\{r \mid W, x\}$ is increasing, where W is the matrix of all network parameters,

$$E\{\Delta w_{ij} \mid W, x\} = a\frac{\partial E\{r \mid W, x\}}{\partial w_{ij}} \tag{2.3}$$

where $a > 0$.

This means that for any REINFORCE algorithm, the expectation of the weight change follows the gradient of the performance measure $E\{r \mid W, x\}$. Therefore, REINFORCE algorithms can be used to perform stochastic maximization of the performance measure.

In the case of the Bernoulli unit with p inputs, the probability p_i is computed as $p_i = f(\sum_{j=1}^{p} w_{ij}x_j)$, where $f(x) = 1/(1 + \exp(-x))$, and it holds that

$$\frac{\partial \ln g_i(y_i; p_i)}{\partial p_i} = \frac{y_i - p_i}{p_i(1 - p_i)}, \tag{2.4}$$

where y_i is the binary output (0 or 1) (Williams, 1992).

2.3 The Reinforcement Clustering Approach. In our approach to clustering based on reinforcement learning (called the RC approach), we consider that each cluster i ($i = 1, \ldots, L$) corresponds to a Bernoulli unit, whose weight vector $w_i = (w_{i1}, \ldots, w_{ip})^\top$ corresponds to the prototype vector for cluster i. At each step, each Bernoulli unit i is fed with a randomly selected pattern x and performs the following operations.

First, the distance $s_i = d(x, w_i)$ is computed, and then the probability p_i is obtained as follows:

$$p_i = h(s_i) = 2(1 - f(s_i)), \tag{2.5}$$

where $f(x) = 1/(1 + \exp(-x))$. Function h provides values in (0, 1) (since $s_i \geq 0$) and is monotonically decreasing. Therefore, the smaller the distance s_i between x and w_i, the higher the probability p_i that the output y_i of the unit will be 1. Thus, when a pattern is presented to the clustering units, they provide output 1 with probability inversely proportional to the distance of the pattern from the cluster prototype. Consequently, the closer (according to some proximity measure) a unit is to input pattern, the higher the probability that the unit will be active (i.e., $y_i = 1$). The probabilities p_i provide a measure of the proximity between patterns and cluster centers. Therefore, if a unit i is active, it is very probable that this unit is close to the input pattern.

According to the immediate reinforcement learning framework, after each cluster unit i has computed the output y_i, the environment (external teacher) must evaluate the decisions by sending a separate reinforcement signal r_i to each unit i. This evaluation is made in such a way that the units update their weights so that the desirable clustering strategy is implemented. In the next section we consider as examples the cases of some well-known clustering strategies.

Following equation 2.2, the use of the REINFORCE algorithm for updating the weights of clustering units suggests that

$$\Delta w_{ij} = a(r_i - b_{ij}) \frac{\partial \ln g_i(y_i; p_i)}{\partial p_i} \frac{\partial p_i}{\partial s_i} \frac{\partial s_i}{\partial w_{ij}}. \tag{2.6}$$

Using equations 2.4 and 2.5, equation 2.6 takes the form

$$\Delta w_{ij} = a(r_i - b_{ij})(y_i - p_i) \frac{\partial s_i}{\partial w_{ij}}, \tag{2.7}$$

which is the weight update equation corresponding to the reinforcement clustering (RC) scheme.

An important characteristic of the above weight update scheme is that it operates toward maximizing the following objective function:

$$R(W) = \sum_{i=1}^{N} \hat{R}(W, x_i) = \sum_{i=1}^{N} \sum_{j=1}^{L} E\{r_j \mid W, x_i\}, \tag{2.8}$$

where $E\{r_j \mid W, x_i\}$ denotes the expected value of the reinforcement received by cluster unit j when the input pattern is x_i.

Consequently the reinforcement clustering scheme can be employed in the case of problems whose objective is the online maximization of a function that can be specified in the form of $R(W)$. The maximization is achieved by performing updates that at each step (assuming input x_i) maximize the term $\hat{R}(W, x_i)$. The latter is valid since from equation 2.3 we have that

$$E\{\Delta w_{kl} \mid W, x_i\} = a \frac{\partial E\{r_k \mid W, x_i\}}{\partial w_{kl}}. \tag{2.9}$$

Since the weight w_{kl} affects only the term $E\{r_k \mid W, x_i\}$ in the definition of $\hat{R}(W, x_i)$, we conclude that

$$E\{\Delta w_{kl} \mid W, x_i\} = a\frac{\partial \hat{R}(W, x_i)}{\partial w_{kl}}. \tag{2.10}$$

Therefore, the RC update algorithm performs online stochastic maximization of the objective function R in the same sense that the LVQ minimizes the objective function J (see equation 2.1) or the online backpropagation algorithm minimizes the well-known mean square error function.

3 The RGCL Algorithm

In the classical LVQ algorithm, only the winning unit $i^\star$ updates its weights, which are moved toward input pattern x, while the weights of the remaining units remain unchanged:

$$\Delta w_{ij} = \begin{cases} a_i(x_j - w_{ij}) & \text{if } i \text{ is the winning unit} \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

For simplicity, in equation 3.1, the dependence of the parameters a_i on time is not stated explicitly. Usually the a_i start from a reasonable initial value and gradually reduce to zero in some way. But in many LVQ implementations (as, for example, in Xu et al., 1993) the parameter a_i remains fixed assuming a small value.

The strategy we would like the system to learn is that when one pattern is presented to the system, only the winning unit (the closest one) becomes active (with high probability) and updates its weights, while the other units remain inactive (again with high probability). To implement this strategy the environment identifies the unit $i^\star$ with maximum p_i and returns a reward signal $r_{i^\star} = 1$ to that unit if it has decided correctly ($y_{i^\star} = 1$) and a penalty signal $r_{i^\star} = -1$ if its guess is wrong ($y_{i^\star} = 0$). The reinforcements sent to the other (nonwinning) units are $r_i = 0$ ($i \neq i^\star$), so that their weights are not affected. Therefore

$$r_i = \begin{cases} 1 & \text{if } i = i^\star \text{ and } y_i = 1 \\ -1 & \text{if } i = i^\star \text{ and } y_i = 0 \\ 0 & \text{if } i \neq i^\star. \end{cases} \tag{3.2}$$

Following this specification of r_i and setting $b_{ij} = 0$ for every i and j, equation 2.7 takes the form

$$\Delta w_{ij} = a r_i (y_i - p_i)\frac{\partial s_i}{\partial w_{ij}}. \tag{3.3}$$

In the case where the Euclidean distance is used ($s_i = d^2(x, w_i) = \sum_{j=1}^{p}(x_j - w_{ij})^2)$), equation 3.3 becomes

$$\Delta w_{ij} = a r_i (y_i - p_i)(x_j - w_{ij}), \tag{3.4}$$

which is the update equation of RGCL.

Moreover, following the specification of r_i (see equation 3.2), it is easy to verify that

$$\Delta w_{ij} = \begin{cases} a|y_i - p_i|(x_j - w_{ij}) & \text{if } i = i^\star \\ 0 & \text{otherwise.} \end{cases} \tag{3.5}$$

Therefore, each iteration of the RGCL clustering algorithm consists of the following steps:

1. Randomly select a sample x from the data set.
2. For $i = 1, \ldots, L$ compute the probability p_i and decide the output y_i of cluster unit i.
3. Specify the winning unit $i^\star$ with $p_{i^\star} = \max_i p_i$.
4. Compute the reinforcements r_i $(i = 1, \ldots, L)$ using equation 3.2.
5. Update the weight vectors w_i $(i = 1, \ldots, L)$ using equation 3.4.

As in the case with the LVQ algorithm, we consider that the parameter a does not depend on time and remains fixed at a specific small value.

The main point in our approach is that we have a learning system that operates in order to maximize the expected reward at the upcoming trial. According to the specification of the rewarding strategy, high values of r are received when the system follows the clustering strategy, while low values are obtained when the system fails in this task. Therefore, the maximization of the expected value of r means that the system is able to follow (on the average) the clustering strategy. Since the clustering strategy aims at minimizing the objective function J, in essence we have obtained an indirect stochastic way to minimize J through the learning of the clustering strategy—that is, through the maximization of the immediate reinforcement r. This intuition is made more clear in the following.

In the case of the RGCL algorithm, the reinforcements are provided by equation 3.2. Using this equation and taking into account that $y_i = 1$ with probability p_i and $y_i = 0$ with probability $1 - p_i$, it is easily to derive from equation 2.8 that the objective function R_1 maximized by RGCL is

$$R_1(X, W) = \sum_{j=1}^{N} \left[p_{i^\star}(x_j) - (1 - p_{i^\star}(x_j)) \right], \tag{3.6}$$

where $p_{i^\star}(x_j)$ is the maximum probability for input x_j. Equation 3.6 gives

$$R_1(X, W) = 2\sum_{j=1}^{N} p_{i^\star}(x_j) - N. \tag{3.7}$$

Since N is a constant and the probability p_i is inversely proportional to the distance $d(x, w_i)$, we conclude that the RGCL algorithm performs updates that minimize the objective function J, since it operates toward the maximization of the objective function R_1.

Also another interesting case results if we set $r_{i^\star} = 0$ when $y_{i^\star} = 0$, which yields the following objective function,

$$R_2(X, W) = \sum_{j=1}^{N} p_{i^\star}(x_j), \tag{3.8}$$

having the same properties as R_1.

In fact the LVQ algorithm can be considered a special case of the RGCL algorithm. This stems from the fact that by setting

$$r_i = \begin{cases} \frac{1}{y_i - p_i} & \text{if } i = i^\star \text{ and } y_i = 1 \\ -\frac{1}{y_i - p_i} & \text{if } i = i^\star \text{ and } y_i = 0, \\ 0 & \text{if } i \neq i^\star \end{cases} \tag{3.9}$$

the update equation, 3.4, becomes exactly the LVQ update equation. Consequently, using equation 2.8, it can be verified that except for minimizing the hard clustering objective function J, the LVQ algorithm operates toward maximizing the objective function:

$$R_3(X, W) = \sum_{j=1}^{N} \left[\frac{p_{i^\star}(x_j)}{1 - p_{i^\star}(x_j)} + \frac{1 - p_{i^\star}(x_j)}{p_{i^\star}(x_j)} \right]. \tag{3.10}$$

Moreover, if we compare the RGCL update equation, 3.5, with the LVQ update equation, we can see that the actual difference lies in the presence of the term $|y_i - p_i|$ in the RGCL update equation. Since y_i may be either one or zero (depending on the p_i value), the absolute value $|y_i - p_i|$ is different (high or low) depending on the outcome y_i. Therefore, under the same conditions (W and x_i), the strength of the weight updates w_{ij} may be different depending on the y_i value. This fact introduces a kind of noise in the weight update equations that assists the learning system to escape from shallow local minima and be more effective than the LVQ algorithm. It must also be stressed that the RGCL scheme is not by any means a global optimization clustering approach. It is a local optimization clustering procedure that exploits randomness to escape from shallow local minima, but it can be trapped in steep local minimum points. In section 4 we present a modification to the RGCL weight update equation that gives the algorithm the property of sustained exploration.

3.1 Other Clustering Strategies. Following the above guidelines, almost every online clustering technique may be considered in the RC framework by appropriately specifying the reinforcement values r_i provided to the clustering units. Such an attempt would introduce the characteristics of "noisy search" in the dynamics of the corresponding technique and would make it more effective in the same sense that the RGCL algorithm seems to be more effective than LVQ according to the experimental tests. Moreover, any distance measure $d(x, w_i)$ may be used provided that the derivative $\partial d(x, w_i)/\partial w_{ij}$ can be computed.

We consider now the specification of the reinforcements r_i to be used in the RC weight update equation, 2.7, in the cases of some well-known online clustering techniques.

3.1.1 Frequency Sensitive Competitive Learning (FSCL) (Ahalt, Krishnamurty, Chen, & Melton, 1990). In the FSCL case it is considered $d(x, w_i) = \gamma_i|x - w_i|^2$ with $\gamma_i = n_i / \sum_j n_j$, where n_i is the number of times that unit i is the winning unit. Also,

$$r_i = \begin{cases} 1 & \text{if } i = i^\star \text{ and } y_i = 1 \\ -1 & \text{if } i = i^\star \text{ and } y_i = 0 \\ 0 & \text{if } i \neq i^\star. \end{cases}$$

3.1.2 Rival Penalized Competitive Learning (RPCL) (Xu et al., 1993). This is a modification of FSCL where the second winning unit i^s moves to the opposite direction with respect to the input vector x. This means that $d(x, w_i) = \gamma_i|x - w_i|^2$ and

$$r_i = \begin{cases} 1 & \text{if } i = i^\star \text{ and } y_i = 1 \\ -1 & \text{if } i = i^\star \text{ and } y_i = 0 \\ -\beta & \text{if } i = i^s \text{ and } y_i = 1 \\ \beta & \text{if } i = i^s \text{ and } y_i = 0 \\ 0 & \text{if } i \neq i^\star \end{cases},$$

where $\beta \ll 1$ according to the specification of RPCL.

3.1.3 Maximum Entropy Clustering (Rose, Gurewitz, & Fox, 1990). The application of the RC technique to the maximum entropy clustering approach suggests that $d(x, w_i) = |x - w_i|^2$ and

$$r_i = \begin{cases} \dfrac{\exp(-\beta|x-w_i|^2)}{\sum_{j=1}^{L}\exp(-\beta|x-w_j|^2)} & \text{if } y_i = 1 \\ -\dfrac{\exp(-\beta|x-w_i|^2)}{\sum_{j=1}^{L}\exp(-\beta|x-w_j|^2)} & \text{if } y_i = 0 \end{cases}$$

where the parameter β gradually increases with time.

3.1.4 Self-Organizing Map (SOM) (Kohonen, 1989). It is also possible to apply the RC technique to the SOM network by using $d(x, w_i) = |x - w_i|^2$ and specifying the reinforcements r_i as follows:

$$r_i = \begin{cases} h_\sigma(i, i^\star) & \text{if } y_i = 1 \\ -h_\sigma(i, i^\star) & \text{if } y_i = 0, \end{cases}$$

where $h_\sigma(i, j)$ is a unimodal function that decreases monotonically with respect to the distance of the two units i and j in the network lattice and σ is a characteristic decay parameter.

4 Sustained Exploration

The RGCL algorithm can be easily adapted in order to obtain the property of sustained exploration. This is a mechanism that gives a search algorithm the ability to escape from local minima through the broadening of the search at certain times (Ackley, 1987; Willams & Peng, 1991). The property of sustained exploration actually emphasizes divergence—return to global searching without completely forgetting what has been learned. The important issue is that such a divergence mechanism is not external to the learning system (as, for example, in the case of multiple restarts); it is an internal mechanism that broadens the search when the learning system tends to settle on a certain state, without any external intervention.

In the case of REINFORCE algorithms with Bernoulli units, sustained exploration is very easily obtained by adding a term $-\eta w_{ij}$ to the weight update equation, 2.2, which takes the form (Williams & Peng, 1991)

$$\Delta w_{ij} = a(r - b_{ij}) \frac{\partial \ln g_i}{\partial w_{ij}} - \eta w_{ij}. \tag{4.1}$$

Consequently the update equation, 3.4, of the RGCL algorithm now takes the form

$$\Delta w_{ij} = a r_i (y_i - p_i)(x_j - w_{ij}) - \eta w_{ij}, \tag{4.2}$$

where r_i is given from equation 3.2. The modification of RGCL that employs the above weight update scheme will be called the SRGCL algorithm (sustained RGCL). The parameter $\eta > 0$ must be much smaller than the parameter a so the term $-\eta w_{ij}$ does not affect the local search properties of the algorithms—that is, the movement toward local minimum states.

It is obvious that the sustained exploration term emphasizes divergence and starts to dominate in the update equations, 4.1 and 4.2, when the algorithm is trapped in local minimum states. In such a case, it holds that the quantity $y_i - p_i$ becomes small, and therefore the first term has negligible contribution. As the search broadens, the difference $y_i - p_i$ tends to become higher, and the first term again starts to dominate over the second term. It must be noted that according to equation 4.2, not only the weights of the winning unit are updated at each step of SRGCL, but also the weights of the units with $r_i = 0$.

The sustained exploration term $-\eta w_{ij}$ can also be added to the LVQ update equation, which takes the form

$$\Delta w_{ij} = \begin{cases} a_i(x_j - w_{ij}) - \eta w_{ij} & \text{if } i \text{ is the winning unit} \\ -\eta w_{ij} & \text{otherwise.} \end{cases} \tag{4.3}$$

The modified algorithm will be called SLVQ (sustained LVQ) and improves the performance of the LVQ in terms of minimizing the clustering objective function J.

Due to the sustained exploration property of SRGCL and SLVQ, they do not converge at local minima of the objective function, since their divergence mechanism allows them to escape from them and continue the exploration of the weight space. Therefore, a criterion must be specified in order to terminate the search, which is usually the specification of a maximum number of steps.

5 Experimental Results

The proposed techniques have been tested using two well-known data sets: the IRIS data set (Anderson, 1935) and the "synthetic" data set used in Ripley (1996). In all experiments the value of $a = 0.001$ was used for LVQ, while for SLVQ we set $a = 0.001$ and $\eta = 0.00001$. For RGCL we have assumed $a = 0.5$ for the first 500 iterations and afterward $a = 0.1$, the same holding for SRGCL where we set $\eta = 0.0001$. These parameter values have been found to lead to best performance for all algorithms. Moreover, the RGCL and LVQ algorithms were run for 1500 iterations and the SRGCL and SLVQ for 4000 iterations, where one iteration corresponds to a single pass through all data samples in arbitrary order. In addition, in order to specify the final solution (with $J_{\min}$) in the case of SRGCL and SLVQ, which do not regularly converge to a final state, we computed the value of J every 10 iterations, and, if it were lower than the current minimum value of J, we saved the weight values of the clustering units.

In previous studies (Williams & Peng, 1991) the effectiveness of stochastic search using reinforcement algorithms has been demonstrated. Nevertheless, in order to compare the effectiveness of RGCL as a randomized clustering technique, we have also implemented the following adaptation of LVQ, called randomized LVQ (RLVQ). At every step of the RLVQ process, each actual distance $d(x, w_i)$ is first modified by adding noise; that is, we compute the quantities $d'(x, w_i) = (1 - n)d(x, w_i)$, where n is uniformly selected in the range $[-L, L]$ (with $0 < L < 1$). A new value of n is drawn for every computation of $d'(x, w_i)$. Then the selection of the winning unit is

Table 1: Average Value of the Objective Function J Corresponding to the Solutions Obtained Using the RGCL, LVQ, RLVQ, SRGCL, and SLVQ Algorithms (IRIS Data Set).

	Average J				
Number of Clusters	RGCL	LVQ	RLVQ	SRGCL	SLVQ
3	98.6	115.5	106.8	86.3	94.5
4	75.5	94.3	87.8	62.5	70.8
5	65.3	71.2	69.4	52.4	60.3

Table 2: Average Value of the Objective Function J Corresponding to the Solutions Obtained Using the RGCL, LVQ, RLVQ, SRGCL, and SLVQ Algorithms (Synthetic Data Set).

	Average J				
Number of Clusters	RGCL	LVQ	RLVQ	SRGCL	SLVQ
4	14.4	15.3	14.8	12.4	13.3
6	12.6	13.7	13.3	10.3	12.3
8	10.3	11.4	10.8	9.2	10.1

done by considering the perturbed values $d'(x, w_i)$, and finally the ordinary LVQ update formula is applied.

Experimental results from the application of all methods are summarized in Tables 1 and 2. The performance of the RLVQ heuristic was very sensitive to the level of the injected noise—the value of L. A high value of L leads to pure random search, while a small value of L makes the behavior of RLVQ similar to the behavior of LVQ. Best results were obtained for $L = 0.35$. We have also tested the case where the algorithm starts with a high initial L value ($L = 0.5$) that gradually decreases to a small final value $L = 0.05$, but no performance improvement was obtained. Finally, it must be stressed that RLVQ may also be adapted in the reinforcement learning framework (in the spirit of subsection 3.1); it can be considered as an extension of RGCL with additional noise injected in the evaluation of the reinforcement signal r_i.

5.1 IRIS Data Set. The IRIS data set is a set of 150 data points in R^4. Each point corresponds to three classes, and there are 50 points of each class in the data set. Of course, the class information is not available during training. When three clusters are considered, the minimum value of the objective function J is $J_{\min} = 78.9$ (Hathaway & Bezdek, 1995) in the case where the Euclidean distance is used.

The IRIS data set contains two distinct clusters, while the third cluster is not distinctly separate from the other two. For this reason, when three

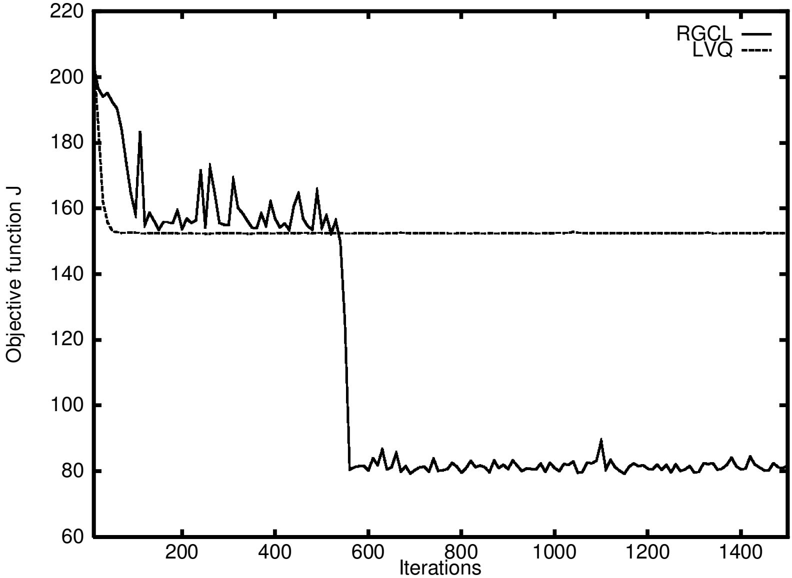

Figure 1: Minimization of the objective function J using the LVQ and the RGCL algorithm for the IRIS problem with three cluster units.

clusters are considered, there is the problem of the flat local minimum (with $J \approx 150$), which corresponds to a solution with two clusters (there exists one dead unit).

Figure 1 displays the minimization of the objective function J in a typical run of the RGCL and LVQ algorithm with three cluster units. Both algorithms started from the same initial weight values. It is apparent that the existence of the local minimum with $J \approx 150$ that corresponds to the solution with two clusters mentioned previously. This is where the LVQ algorithm is trapped. On the other hand, the RGCL algorithm manages to escape from the local minimum and oscillate near the global minimum value $J = 78.9$.

We have examined the cases of three, four, and five cluster units. In each case, a series of 20 experiments was conducted. In each experiment the LVQ, RGCL, and RLVQ algorithms were tested starting from the same weight values that were randomly specified. Table 1 presents the average values (over the 20 runs) of the objective function J corresponding to the solutions obtained using each algorithm. In all cases, the RGCL algorithm is more effective compared to the LVQ algorithm. As expected, the RLVQ algorithm was in all experiments at least as effective as the LVQ, but its average performance is inferior compared to RGCL. This means that the injection of noise during prototype selection was sometimes helpful and assisted the LVQ algorithm to achieve better solutions, while in other experiments it had no effect on LVQ performance.

Table 1 also presents results concerning the same series of experiments (using the same initial weights) for SRGCL and SLVQ. It is clear that significant improvement is obtained by using the SLVQ algorithm in place of LVQ, as well as that the SRGCL is more effective than SLVQ and, as expected, it also improves the RGCL algorithm. On the other hand, the sustained versions require a greater number of iterations.

5.2 Synthetic Data Set. The same observations were verified in a second series of experiments where the synthetic data set is used. In this data set (Ripley, 1996), the patterns are two-dimensional, and there are two classes, each having a bimodal distribution; thus, there are four clusters with small overlaps. We have used the 250 patterns that are considered in Ripley (1996) as the training set, and we make no use of class information.

Several experiments have been conducted on this data set concerning the RGCL, LVQ, and RLVQ algorithms first and then SRGCL and SLVQ. The experiments were performed assuming four, six, and eight cluster units. In analogy with the IRIS data set, for each number of clusters, a series of 20 experiments were performed (with different initial weights). In each experiment, each of the four algorithms is applied with the same initial weight values. The obtained results concerning the average value of J corresponding to the solutions provided by each method are summarized in Table 2. Comparative performance results are similar to those obtained with the IRIS data set.

A typical run of the RGCL and LVQ algorithms with four cluster units starting from the same positions (far from the optimal) is depicted in Figures 2 and 3, respectively. These figures display the data set (represented with crosses), as well as the traces of the four cluster units until they reach their final positions (represented with squares). It is clear that the RGCL provides a four-cluster optimal solution (with $J = 12.4$), while the LVQ algorithm provides a three-cluster solution with $J = 17.1$. The existence of a dead unit at position $(-1.9, -0.55)$ (square at the low left corner of Figure 3) in the LVQ solution and the effect of randomness on the RGCL traces that supplies the algorithm with better exploration capabilities are easily observed.

Moreover, in order to perform a more reliable comparison between the RGCL and RLVQ algorithms, we have conducted an additional series of experiments on the synthetic data set assuming four, six, and eight cluster units. For each number of clusters we have conducted 100 runs with each algorithm in exactly the same manner with the previous experiments. Tables 3 and 4 display the statistics of the final value of J obtained with each algorithm, and Table 5 displays the percentage of runs for which the performance of RGCL was superior ($J_{RGCL} < J_{RLVQ}$), similar ($J_{RGCL} \sim J_{RLVQ}$), or inferior to RLVQ ($J_{RGCL} > J_{RLVQ}$). More specifically, the performance of the RGCL algorithm with respect to RLVQ was superior when

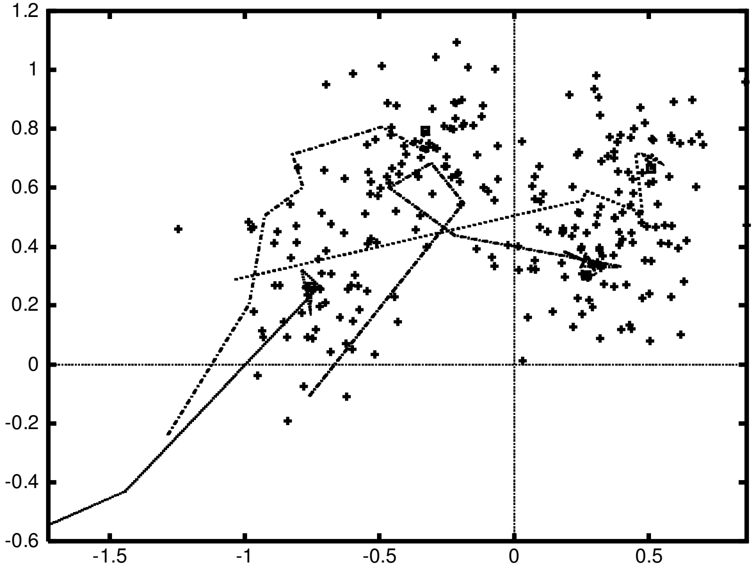

Figure 2: Synthetic data set and traces of the four cluster prototypes corresponding to a run of the RGCL algorithm with four cluster units (four traces).

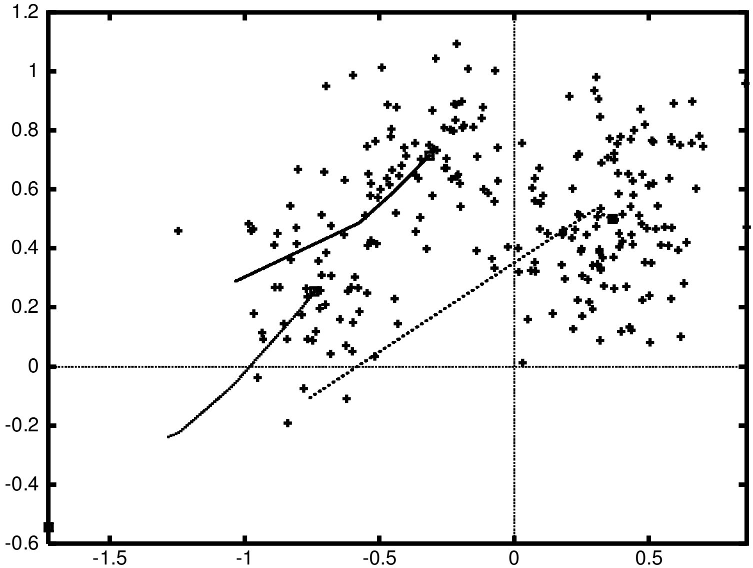

Figure 3: Synthetic data set and traces of the cluster prototypes corresponding to a run of the LVQ algorithm with four cluster units (three traces and one dead unit).

Table 3: Statistics of the Objective Function J Corresponding to Solutions Obtained from 100 Runs with the RGCL Algorithm, Synthetic Data Set.

	J			
Number of Clusters	Average	Standard	Best	Worst
4	14.5	2.2	12.4	28.9
6	12.4	1.8	9.1	17.2
8	10.2	2.5	7.1	17.2

Table 4: Statistics of the Objective Function J Corresponding to Solutions Obtained from 100 Runs with the RLVQ Algorithm, Synthetic Data Set.

	J			
Number of Clusters	Average	Standard	Best	Worst
4	15.1	3.1	12.4	28.9
6	13.3	3.7	9.1	28.9
8	10.7	4.2	7.5	28.9

$J_{RGCL} < J_{RLVQ} - 0.3$, similar when $|J_{RGCL} - J_{RLVQ}| \leq 0.3$, and inferior when $J_{RGCL} > J_{RLVQ} + 0.3$. The displayed results make clear the superiority of the RGCL approach, which not only provides solutions that are almost always better or similar to RLVQ but also leads to solutions that are more reliable and consistent, as indicated by the significantly lower values of the standard deviation measure.

6 Conclusion

We have proposed reinforcement clustering as a reinforcement-based technique for online clustering. This approach can be combined with any online clustering algorithm based on competitive learning and introduces a degree of randomness to the weight update equations that has a positive effect on clustering performance.

Further research will be directed to the application of the approach to

Table 5: Percentage of Runs for Which the Performance of the RGCL Algorithm was Superior, Similar, or Inferior to RLVQ.

Number of Clusters	$J_{RGCL} < J_{RLVQ}$	$J_{RGCL} \sim J_{RLVQ}$	$J_{RGCL} > J_{RLVQ}$
4	47%	51%	2%
6	52	45	3
8	64	34	2

clustering algorithms other than LVQ, for example, the ones that are reported in section 3. The assessment of the performance of those algorithms under the RC framework needs to be examined and assessed. Moreover, the application of the proposed technique to real-world clustering problems (for example, image segmentation) constitutes another important future objective.

Another interesting direction concerns the application of reinforcement algorithms to mixture density problems. In this case, the employment of doubly stochastic units—those with a normal component followed by a Bernoulli component—seems appropriate (Kontoravdis, Likas, & Stafylopatis, 1995). Also of great interest is the possible application of the RC approach to fuzzy clustering, as well as the development of suitable criteria for inserting, deleting, splitting, and merging cluster units.

Acknowledgments

The author would like to thank the anonymous referees for their useful comments and for suggesting the RLVQ algorithm for comparison purposes.

References

Ackley, D. E. (1987). *A Cconnectionist machine for genetic hillclimbing*. Norwell, MA: Kluwer.

Ahalt, S. C., Krishnamurty, A. K., Chen, P., & Melton, D. E. (1990). Competitive learning algorithms for vector quantization. *Neural Networks*, *3*, 277–291.

Anderson, E. (1935). The IRISes of the Gaspe Peninsula. *Bull. Amer. IRIS Soc.*, *59*, 381–406.

Barto, A. G., & Anandan, P. (1985). Pattern recognizing stochastic learning automata. *IEEE Transactions on Systems, Man and Cybernetics*, *15*, 360–375.

Hathaway, R. J., & Bezdek, J. C. (1995). Optimization of clustering criteria by reformulation. *IEEE Trans. on Fuzzy Systems*, *3*, 241–245.

Kaelbling, L., Littman, M., & Moore, A. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.

Kohonen, T. (1989). *Self-organization and associative memory* (3rd ed.). Berlin: Springer-Verlag.

Kontoravdis, D., Likas, A., & Stafylopatis, A. (1995). Enhancing stochasticity in reinforcement learning schemes: Application to the exploration of binary domains. *Journal of Intelligent Systems*, *5*, 49–77.

Martinez, T., Berkovich, S., & Schulten, G. (1993). "Neural-gas" network for vector quantization and its application to time-series prediction. *IEEE Trans. on Neural Networks*, *4*, 558–569.

Ripley, B. (1996). *Pattern recognition and neural networks*. Cambridge: Cambridge University Press.

Rose, K., Gurewitz, F., & Fox, G. (1990). Statistical mechanics and phase transitions in clustering. *Physical Rev. Lett.*, *65*, 945–948.

Williams, R. J. (1988). *Toward a theory of reinforcement learning vonnectionist dystems* (Tech. Rep. NU-CCS-88-3). Boston, MA: Northeastern University.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*, 229–256.

Williams, R. J., & Peng, J. (1991). Function optimization using connectionnist reinforcement learning networks. *Connection Science*, *3*, 241–268.

Xu, L., Krzyzak, A., & Oja, E. (1993). Rival penalized competitive learning for clustering analysis, RBF net, and curve detection. *IEEE Trans. on Neural Networks*, *4*, 636–649.

Received February 26, 1998; accepted November 24, 1998.