A new frequency sensitive competitive learning algorithm for data clustering

Sagar Sen

C/o Supercomputer Education and Research Centre, Indian Institute of Science, Bangalore 560012
Email: sagarsen@rediffmail.com

ABSTRACT

Competitive learning algorithms adaptively compute the cluster centers in a dataset with clustered information. We investigate two such competitive learning algorithms that are primarily based on frequency sensitivity. Frequency sensitive competitive learning (FSCL) and Rival penalized competitive learning (RPCL) have interesting properties when applied on a dataset concurrently with equivalent weight initialisations. These algorithms when run concurrently tend to separate out the redundant prototypes based on a distance measure. The distance measure helps us distinguish between true cluster centers from redundant centers. The property of RPCL to push a redundant unit (which acts as the rival to a legitimate cluster center) far away from the corresponding FSCL unit (the legitimate cluster center) gives us the required distance measure. We have applied this algorithm to image segmentation apart from regular clustering analysis.

1. INTRODUCTION

Statistical datasets from several experiments often possess inherent structure and subtle important features. Data clustering aims at discovering and emphasizing structure that is hidden in a data set. A certain number of representatives (prototypes, cluster centers or units) encapsulate the essential information conveyed by the nature of the clusters in the data. Competitive learning is an adaptive strategy to find cluster centers. Competitive learning is unsupervised learning in nature, i.e., we try to minimize the Euclidean distance between a cluster center and the adjacent points in the dataset based on a learning rate instead of teaching by specifying input/output pairs. Competitive learning algorithms must attack several problems associated with clustering analysis. Some of the important problems are

P1. Under-utilization problem or dead nodes problem Due to inappropriate initialisation of some cluster prototypes they never become the winner (winner is the prototype that has the smallest distance from the input data point), therefore have no contribution to learning.

P2. Automatically finding the number of clusters and the respective cluster centers The very act of random initialisation of prototypes makes it impossible for us to ascertain the convergence of cluster centers to the respective clusters. In all probability there might be a cluster that remains undetected by the prototypes.

P3. Multiple prototypes per cluster This problem arises when one prototype wins the competition repeatedly. The other prototypes are deprived. This can partly be solved by making the competition sensitive to the frequency of winning of each prototype. The less frequent winners get a chance to win a point and hence distribute the load within all the prototypes.

Several approximate techniques for competitive learning exist that try to solve the above problem in many intuitive ways. Due to the statistical nature of data it is impossible for anyone to claim that any of the algorithms in the literature give optimal results.

We analyze two important algorithms from the literature – Frequency Sensitive Competitive Learning (FSCL) [3] and Rival Penalized Competitive Learning (RPCL) [2]. In Section 2 we discuss these algorithms. Following this analysis we elucidate subtle features in these algorithms that help us counter inherent problems in clustering analysis. We introduce Concurrent Competitive Learning (CCL) in Section 3, which is based on FSCL and RPCL. In

Section 4 results of experimentation of CCL with data clustering and image segmentation problems is shown. We conclude in Section 5.

2. TWO COMPETITIVE LEARNING ALGORITHMS

Frequency sensitive competitive learning (FSCL) [3] and rival penalized competitive learning (RPCL) [2] algorithms are introduced in this section. We reiterate the fact, from the original literature, that they try to solve problem P1 and P3 specified in Section 1 of the paper. Incorporating some history/frequency sensitivity into the competitive learning rule provides a way to alleviate the problem of totally unlearned neurons or prejudiced training. There are two approaches:

1. Modulate the selection of a winner by the frequency sensitivity.
2. Modulate the learning rate by the frequency sensitivity.

In Table 2.1 we list of the essential parameters and variable used in both FSCL and RPCL. In FSCL he rate of training can also be modulated by frequency sensitivity. Consider a dataset D.

Table 2.1 Parameters used in FSCL and RPCL

k	It is initial estimation of the number of clusters in the given data
u_i $1\le i \le k$	The output units or the output vector of dimension k
$\mathbf{w_i}$ $1\le i \le k$	k - weight vectors each of dimension d
$\mathbf{x_i}$ $1\le i \le d$	The d-dimensional input vector from data set D
$0\le\alpha_c\le1$	Learning rate for winner
$0\le\alpha_r\le1$	Learning rate for rival
$\mathbf{w_c}$	d-dimensional weight vector corresponding to the winner
$\mathbf{w_r}$	d-dimensional weight vector corresponding to the rival
n_i $1\le i \le k$	Cumulative number of occurrences of $u_i = 1$
$\gamma_j = n_j / \sum_{i=1}^{k} n_i$	The conscience factor to reduce the winning rate of constant winners

Detailed analysis for FSCL can be found in [3]. The most important observations one can make regarding FSCL are

- **All prototypes have equal chance** The competition is based on the conscience factor which in turn is a function of the frequency of winning. Deprived prototypes will have lower conscience compared to the frequent winners. The lower conscience factor will allow the algorithm to push the infrequent winner to win more frequently. Hence, a balance is maintained and all the prototypes get a reasonably equal share of the input data.
- **Prototypes always win** Since there is no de-learning involved (though there is a possibility of several prototypes occupying the same cluster) there is no chance for the prototypes to wander away from the cluster regions. This property is pivotal in the CCL algorithm.

As an example, we present the following competitive learning rule:
Function FSCL (D,k)

Step 1: Randomly take a sample x from a data set D; for i=1,…, k, let

$$u_i = \begin{cases} 1, \text{ if } i=c \text{ such that } \gamma_c \|\mathbf{x} - \mathbf{w_c}\|^2 = \min_i \gamma_i \|\mathbf{x} - \mathbf{w_i}\|^2, \\ 0, \text{ otherwise.} \end{cases} \qquad (1a)$$

Step 2: Update the weight vector w_i by

$$\Delta \mathbf{w_i} = \begin{cases} \alpha_c (\mathbf{x} - \mathbf{w}_i), \text{ if } u_i = 1 \\ 0, \text{ otherwise} \end{cases} \tag{1b}$$

FSCL is widely used and is a very simple algorithm. The most important problem with FSCL is its inability to find exact number of cluster centers.

RPCL [2] is a clever extension of the FSCL algorithm. The basic idea of frequency sensitivity has been preserved. In addition de-learning is introduced in the rivals (second best to the winner). This improvement in the algorithm automatically makes one prototype occupy one cluster. The rivals to a prototype are pushed away from the cluster it is trying to occupy, thereby becoming the only prototype to learn from the cluster. The rival penalization in turn implicitly pushes the rivals towards other clusters. Finally all the extra units are pushed far away from the clusters that already have a center. These extra units can be used later on if new data has to be trained. The algorithmic description for RPCL is

Function RPCL (D, k)

Step 1: Randomly take a sample x from a data set D, for i=1,…, k, let

$$u_i = \begin{cases} 1, \text{ if } i=c \text{ such that } \gamma_c \|\mathbf{x} - \mathbf{w_c}\|^2 = \min_i \gamma_i \| \mathbf{x} - \mathbf{w_i}\|^2, \\ -1, \text{ if } i=r \text{ such that } \gamma_r \|\mathbf{x} - \mathbf{w_r}\|^2 = \min_{i\neq c} \gamma_i \| \mathbf{x} - \mathbf{w_i} \|^2, \\ 0, \text{ otherwise.} \end{cases} \tag{2a}$$

Step 2: Update the weight vector w_i by

$$\Delta \mathbf{w_i} = \begin{cases} \alpha_c (\mathbf{x} - \mathbf{w_i}), \text{ if } u_i = 1 \\ -\alpha_r (\mathbf{x} - \mathbf{w_i}), \text{ if } u_i = -1 \\ 0, \text{ otherwise} \end{cases} \tag{2b}$$

The feature of RPCL to push the redundant units away from the center will provide important basis for CCL to compare FSCL and RPCL results.

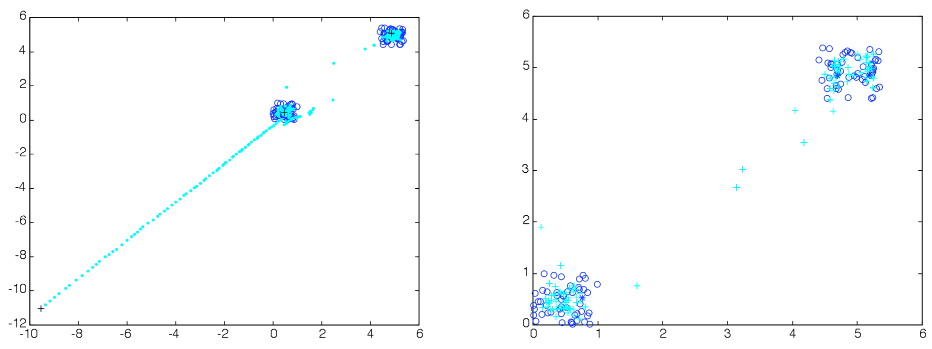


Fig.1 RPCL results (left) on a dataset with two clusters and 3 prototypes. Observe the rival unit being pushed far away. FSCL results with same conditions. All centers are close to the clusters.

3. CONCURRENT COMPETITIVE LEARNING

We showed how FSCL and RPCL solve problems P1 and P3 introduced in Section 1. In this section we develop an algorithm that attempts to tackle problem P2 from Section 1 (the problem of automatically finding the exact number of cluster centers). We can reiterate two important properties from FSCL and RPCL that will be the basis for CCL[♣].

- In FSCL all the cluster centers occupy regions close to the data points. Both the representatives and redundant centers are in close proximity.
- Cluster centers in RPCL are de-learned when they act as rivals to the winner of a cluster. This way the redundant units are pushed far away from the proximity of the data set and other centers.

We run FSCL and RPCL concurrently with the same random initialisation for weight vectors $\mathbf{w}^f$ and $\mathbf{w}^r$ each corresponding to the FSCL weight vector and the RPCL weight vector respectively. If the weight vector $\mathbf{w}^f$ is a representative of a center and so is w^r then both are very close to each other. However, if $\mathbf{w}^f$ and $\mathbf{w}^r$ correspond to redundant centers both are pushed far away from each other and hence must be removed from the set of representative cluster centers. This process continues iteratively to weed out all the redundant centers and ultimately keep only the representative cluster centers.

The distance between vector $\mathbf{w}^r$ and $\mathbf{w}^f$ must be above a certain threshold to be removed from the set of representative cluster centers. The aim of a competitive learning algorithm is to obtain cluster centers tuned to a particular region of input data. The concept of threshold for a cluster center is borrowed from [4]. The idea here is to check whether $\mathbf{w}^r$ is within the hypersphere with the threshold as the radius measure and center $\mathbf{w}^f$. If $\mathbf{w}^r$ is outside this hypersphere we conclude that it is redundant and remove it from the set of representative cluster centers.

We enhance the general form of our activation function u_i from equation (1a) from Section 2

$$u_i = \begin{cases} 1, \text{ if } i=c \text{ such that } (T_i^f - \gamma_c \|\mathbf{x} - \mathbf{w}^f_c\|^2) = \min_i (T_i^f - \gamma_i \|\mathbf{x} - \mathbf{w}^f_i\|^2), \\ 0, \text{ otherwise.} \end{cases} \tag{3a}$$

where T_i^f is the threshold for the ith prototype in FSCL.

Also we make an equivalent change in the activation function from equation (2a) from Section 2

$$u_i = \begin{cases} 1, \text{ if } i=c \text{ such that } T_i^r - \gamma_c \|\mathbf{x} - \mathbf{w}^r_c\|^2 = \min_i (T_i^r - \gamma_i \|\mathbf{x} - \mathbf{w}^r_i\|^2), \\ -1, \text{ if } i=r \text{ such that } T_i^r - \gamma_r \|\mathbf{x} - \mathbf{w}^r_r\|^2 = \min_{i\neq c} (T_i^r - \gamma_i \|\mathbf{x} - \mathbf{w}^r_i\|^2), \\ 0, \text{ otherwise.} \end{cases} \tag{3b}$$

where T_i^r is the threshold for the ith prototype in RPCL.

The thresholds are modified based on the following rule of standard learning [4]:

$T^f_i(t+1)=T^f_i(t)+\Delta T^f_i(t)$

[♣] Note: All vectors are in bold. The variables with superscript f correspond to the variables for FSCL. The variables with superscript r correspond to the variables for RPCL.

$\Delta T^f_i(t)= \alpha^f_i \, (||\mathbf{x} - \mathbf{w_i}|| - T^f_i(t))$

$T^r_i(t+1)=T^r_i(t)+\Delta T^r_i(t)$

$\Delta T^r_i(t)= \alpha^r_i \, (||\mathbf{x} - \mathbf{w_i}|| - T^r_i(t))$

This dynamic change in the threshold values prevents a cluster center from affecting other clusters outside its region of operation.

The weight units start with a random initialisation. The thresholds and the learning rates are initialised.

Function CCL (D, k, TOL)

Step 1. Randomly take a sample x from the data set D, for i=1…k

$$u^f_i = \begin{cases} 1, \text{ if } i=c \text{ such that } (T^f_i - \gamma_c \, ||\mathbf{x} - \mathbf{w^f_c}||^2) = \min_i (T^f_i - \gamma_i \, || \mathbf{x} - \mathbf{w^f_i} ||^2), \\ 0, \text{ otherwise.} \end{cases} \quad (4a)$$

$$u^r_i = \begin{cases} 1, \text{ if } i=c \text{ such that } T^r_i - \gamma_c \, ||\mathbf{x} - \mathbf{w^r_c}||^2 = \min_i (T^r_i - \gamma_i \, || \mathbf{x} - \mathbf{w^r_i}||^2), \\ -1, \text{ if } i=r \text{ such that } T^r_i - \gamma_r \, ||\mathbf{x} - \mathbf{w^r_r}||^2 = \min_{i \neq c} (T^r_i - \gamma_i \, || \mathbf{x} - \mathbf{w^r_i} ||^2), \\ 0, \text{ otherwise.} \end{cases} \quad (4b)$$

Step 2. Update the weight vectors $\mathbf{w^f_i}$ and $\mathbf{w^r_i}$ adopting the modified competitive law

$$\mathbf{\Delta w^f_i} = \begin{cases} \alpha^f_i \, (\mathbf{x} - \mathbf{w^f_i}), \text{ if } u^f_i = 1 \\ 0, \text{ otherwise} \end{cases} \quad (4c)$$

$\mathbf{w^f_i = w^f_i + \Delta w^f_i}$

$$\mathbf{\Delta w^r_i} = \begin{cases} \alpha^r_{ci} \, (\mathbf{x} - \mathbf{w^r_i}), \text{ if } u^r_i = 1 \\ -\alpha^r_{ri} \, (\mathbf{x} - \mathbf{w^r_i}), \text{ if } u^r_i = -1 \\ 0, \text{ otherwise} \end{cases} \quad (4d)$$

$\mathbf{w^r_i = w^r_i + \Delta w^r_i}$

Step 3. Update the learning coefficients (4e-g)

$$\begin{aligned} \alpha^f_i &= \alpha^f_i \, e^{-0.01} \\ \alpha^r_{ci} &= \alpha^r_{ci} \, e^{-0.01} \\ -\alpha^r_{ri} &= -\alpha^r_{ri} \, e^{-0.01} \end{aligned}$$

Step 4. Update the threshold:

$$T^f_i = T^f_i(t) + \Delta T^f_i$$

$$\Delta T^f_i = \alpha^f_i\ (\ \|\mathbf{x} - \mathbf{w}_i\| - T^f_i\ ) \qquad (4h\text{-}k)$$

$$T^r_i T^r_i(t) + \Delta T^r_i$$

$$\Delta T^r_i = \alpha^r_i\ (\ \|\mathbf{x} - \mathbf{w}_i\| - T^r_i)$$

Step 5. Compute the distance and check

```
If $\| \mathbf{w}^r_i - \mathbf{w}^f_i \| > T^f_i$ then

        Remove ith weight vectors $\mathbf{w}^r_i$ and $\mathbf{w}^f_i$
        k=k-1;
end
```

Step 6. Iterate to step1 until

$$\alpha^f_i \le \varepsilon$$

where ε is a constant coefficient ($\varepsilon = 0.01$).

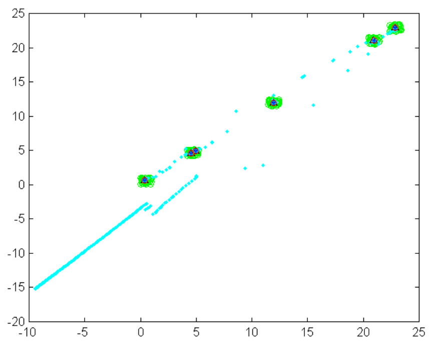
4. EXPERIMENTS

In order to show the capability of CCL to find the correct number of cluster centers we generated random datasets in two-dimensional space with each cluster distributed in the sense of Gaussian.

Sample dataset:

Number of data points	200
Exact number of clusters	5
Initial estimate of the number of clusters	7
Detected number of clusters	6
Initial threshold T^f and T^r	$0.5\text{x}10^{-4}$
Initial learning rate α_c	0.5
Initial learning rate α_r	-0.02

For simplicity we consider only a single threshold values for all elements and also single learning rate for all elements. This alleviation over the number of parameters has been done due to the low complexity of the test data. By making use of all the parameters we can very accurate results however the process of finding them will be slower.



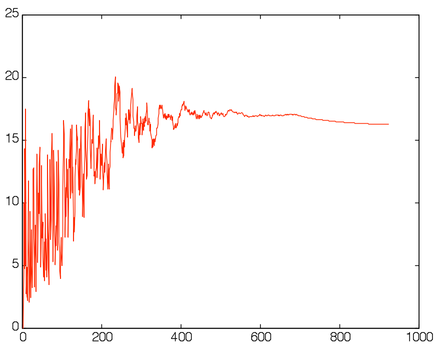


Fig. 2 (a) CCL output with detection of 6 cluster centers. (b) Variation of threshold until constant value is obtained

Next, we consider the problem of image segmentation. Image segmentation is the process of grouping similar regions in a figure. 8-bit Greyscale images were considered for the segmentation tasks. The test images are shown in Fig. 3.

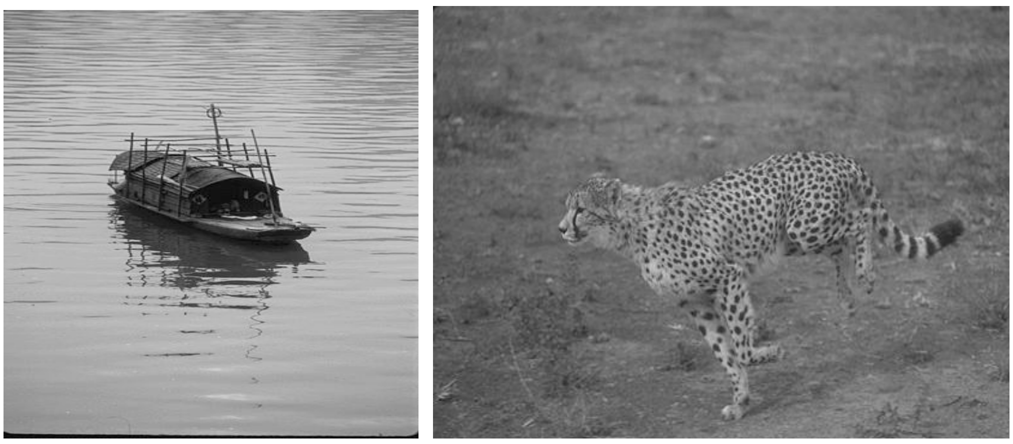


Fig. 3. (a) and (b) Test images for image segmentation task.

For the image segmentation task we made initial estimates of 7 segments and took only about 150 points in random. This is contrary to regular segmentation algorithms like thresholding-based techniques that need to process every pixel for accurate segmentation.

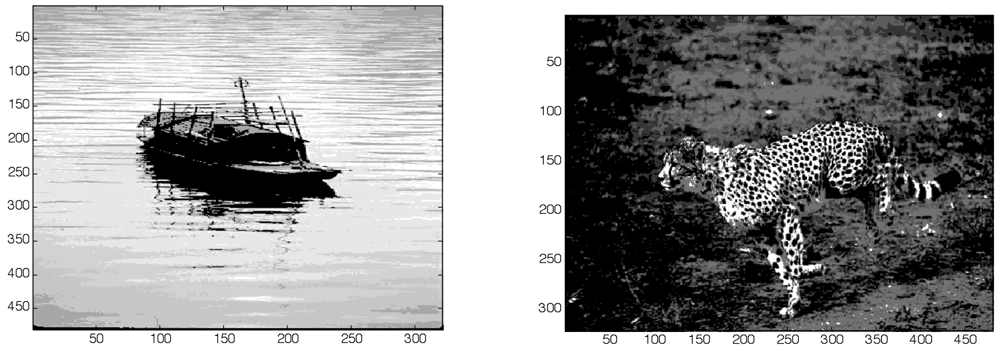


Fig. 4. (a) and (b) Results of segmentation on Fig. 3(a) and Fig. 3(b)

The number of regions detected by CCL for Fig. 3(a) was 6 and the number of regions detected by CCL for Fig. 3(b) was 5. Clearly we can see the compression that CCL can provide by automatically determining the correct number of cluster centers.

6. CONCLUSION

Concurrent competitive learning is different from other competitive learning algorithm in the sense of its mechanism to automatically determine the number of cluster centers. However the most important drawback in CCL is the need to store twice the number of weights during the learning phase. Once the learning phase is over only the pertinent weight vectors may be stored as feature vectors for classification tasks.

The inherently parallel nature of CCL makes it an ideal candidate for hardware implementation and implementation over distributed systems to solve large-scale problems.

Note: MATLAB 5.3 version of CCL may be obtained from the author. Email the author at sagarsen@rediffmail.com

REFERENCES

[1] S.C. Ahalt, A.K. Krishnamurthy, P. Chen, and D.E. Melton, “Comeptitive learning algorithms for vector quantization,” Neural Networks vol. 3, pp. 277-291, 1990

[2] Lei Xu, Adam Krzyzak, and Erkki Oja, “Rival Penalized Competitive Learning for Clustering Analysis, RBF Net, and Curve detection.”, IEEE Trans. Neural Networks vol. 4 No. 4 July 1993

[3] DeSieno, D. “Adding a conscience to competitive learning”. Proceedings of the International Conference on Neural Networks, 117-124, 1988

[4] Giuseppe Acciani et al. “A feature extraction unsupervised neural network for an environmental data set.”, Neural Networks, vol. 16 (special issue), 427-436, 2003