A NEW COMPETITIVE LEARNING ALGORITHM FOR DATA CLUSTERING

Corina Botoca[1], Georgeta Budura[2], Nicolae Miclau[3]

*Communication Department, "Politehnica" University of Timisoara, Romania

corina.botoca@etc.utt.ro1; georgeta.budura@etc.utt.ro; nicolae.miclau@etc.utt.ro

Abstract This paper presents a new competitive learning algorithm for data clustering, named the dynamically penalized rival competitive learning algorithm (DPRCA). It is a variant of the rival penalized competitive algorithm [1] and it performs appropriate clustering without knowing the clusters number, by automatically driving extra seed points far away from the input data set. It doesn't have the "dead neurons" problem. The performances of the DPRCA algorithm were tested by simulations carried out considering different conditions of noise.

Key words: competitive learning algorithms, neural networks

1. INTRODUCTION

Competitive learning is an efficient tool for data clustering, widely applied in a variety of signal processing problems such as data compression, classification, adaptive noise cancellation pattern recognition and image processing [2].

The typical competitive learning algorithm, named *k*-means algorithm [3], partitions the input data set into *k* categories (called clusters) each represented finally by its centre, that adaptively change, starting from some initial values named seed points. The major drawbacks of the *k*-means algorithm are that it needs to know the exact number of clusters *k* and that it has the "dead neurons" problem. This means that if a centre is inappropriately chosen, it may never be updated, thus it may never represent a category.

To circumvent the "dead neurons" problem it was proposed an extension of the k-means algorithm, named the frequency sensitive competitive learning (FSCL) [4]. The FSCL algorithm reduces the learning rate of the frequent winners, so their chance to win, strategy called also "with conscience". Although it can almost successfully assign one or more seed points to a cluster without the "dead neurons" problem, it also needs knowing the exact number of clusters.

The rival penalized competitive learning (RPCL) algorithm [1] performs appropriate clustering without knowing the clusters number and eliminates the "dead neurons" problem. The RPCL algorithm rewards the winning neuron and penalizes with a de-learning rate the second winner, named rival. The algorithm is quite simple and provides a better convergence than the *k*-means and FSCL algorithms, but is rather sensitive to the selection of the de-learning rate. To eliminate this drawback, we introduce in this paper a new competitive algorithm, the dynamically penalized rival competitive learning algorithm. The DRPCA algorithm circumvents the selection problem of the de-learning rate by always fixing it at the same value as the learning rate and dynamically controlling it. If the first rival distance to the winner is closer than the one between the winner and the input, the rival will be more penalized; otherwise the penalization will be gradually attenuated as the distance between the winner and the rival increases. The DRPCA algorithm also drives away the extra number of seed points.

2. EXPERIMENTAL

The DPRCA algorithm calculates a distance, between the input and the centres. The distance may be of different types, but here the Euclidian norm is used. The centre j with the minimum distance is declared winner:

$$j = arg\ min\left(\gamma_i \left\| x(n) - c_i(n) \right\|\right), \quad i = \overline{1, n_h} \tag{1}$$

where $x(n)$ is the input vector, c_i is the centre vector (seed point) of neuron i, n_h is the number of the neurons and γ_i is the relative winning frequency of the centre c_i, defined as:

$$\gamma_i = \frac{s_i}{\sum_{i=1}^{n_h} s_i} \tag{2}$$

where s_i is the number of times when the centre i has won the competition in the past. The relative winning frequency γ_i, solves the "dead neuron" problem. The neurons that have won the competition during the past have a reduced chance to win again, proportional with their frequency. As the RPCL algorithm [1], whose variant is, the DRPCA determines not only the winning neuron, but also the second winning neuron r, named rival, with the relation:

$$r = arg\ min\big|_{i \neq j}\left(\gamma_i \left\| x(n) - c_i(n) \right\|\right), \quad i = \overline{1, n_h} \tag{3}$$

The winning neuron centre is moved with a fraction η towards the input. The second winning neuron will move away from the input its centre, with a ratio β, called the de-learning rate. All the others neurons will not change their centre vector. So the learning law can be synthesized in the following relation:

$$c_i(n+1)=\begin{cases} c_i(n)+\eta\,[x(n)-c_i(n)] & if\ i=j \\ c_i(n)-\beta[x(n)-c_i(n)] & if\ i=r \\ c_i(n) & if\ i\neq j \quad and \quad i\neq r \end{cases} \tag{4}$$

Comparative to RPCL, the DRPCA algorithm applies a new mechanism to control the rival penalization. For this it introduces a new factor, the penalization strength:

$$p(c_i(n))=\frac{min(\|x(n)-c_w\|,\|c_w-c_r\|}{\|c_w-c_r\|} \tag{5}$$

where c_w and c_r are the centre of the winner, respectively of the rival. With this factor the de-learning rate β in equation (4) becomes:

$$\beta=-\eta\,p(c_i(n)) \tag{6}$$

It can be noticed that if $\|x(n)-c_w\|\leq\|c_w-c_r\|$, than the rival will be fully penalized with the learning rate η. Otherwise, the rival will be penalized with the de-learning rate $\eta\ p(c_i(n)$, which is gradually attenuated as the distance between the winner and its rival increases. So, the DRPCA algorithm is actually a generalization of the RPCL.

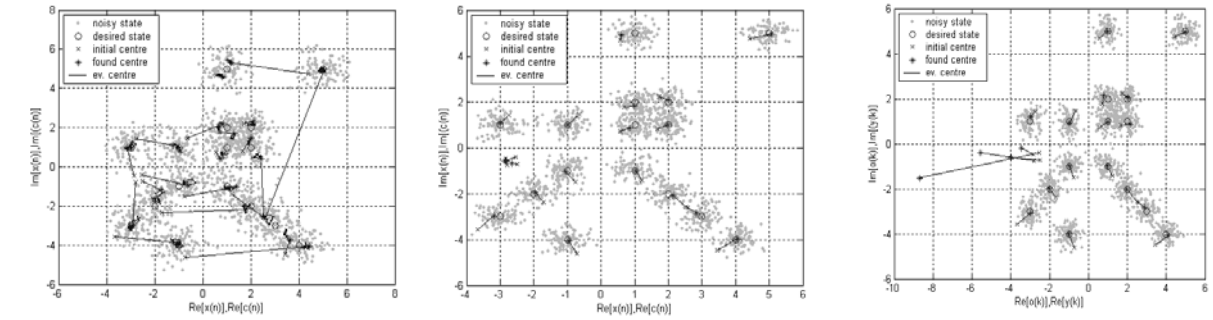
3. RESULTS AND DISCUSSIONS

We have trained a complex radial basis function neural network [1], [2], with the FSCL, RPCL and the new DRPCA algorithm. Different numbers of clusters and numbers of seed point were used in our experiments. Complex input signals were generated using a white Gaussian noise, around the desired centres clusters. Experiments were done for different noise dispersions. The RBF centres were randomly initialized to a subset of the input data signals. In all cases the DRPCA algorithm could find the desired centres clusters, had driven away the extra number of seed points and had a better convergence than the other two algorithms.

Figures 1, 2 and 3 represent the simulations results after 30 training epochs, in case of 16 desired states , 1600 noisy input states x(n), for a SNR=13dB, 20 initial and 20 final positions of

the RBF centres, for the FSCL, RPCL and DRPCA algorithms. The following learning constants were used: η=0.001 and for the RPCL algorithm β=0.0001.

The FSCL algorithm cann't deal with the extra number of seed points. Both other algorithms, RPCL and DRPCA succeed to orientate the RBF centres to the desired states. As one can observe the DRPCA algorithm drives away the extra number of seed points and can find closer positions of RBF centres to the desired states than the RPCL algorithm, so its convergence is better.



a) b) c)

Fig.1 The desired states , the input noisy states *x(n)*, the initial and final positions of the RBF centres in case of a *SNR*=13dB, after 30 training epochs, using: a)FSCL algorithm; b) RPCA algorithm; c)DRPCA algorithm

CONCLUSIONS

The proposed competitive algorithm, namely DRPCA, while training the centres of the RBF network, rewards the winner and dynamically penalizes its closest rival. In comparison with the classic k-means algorithm, it doesn't have the "dead neurons" problem. If compared with the FSCL and the RPCL algorithms, it has a better and faster convergence. So the RPCL algorithm is adequate to the adaptive clustering of fast varying signals corrupted by noise.

REFERENCES

[1] N.Miclau , C. Botoca, G. Budura, *Nonlinear Complex Channel Equalization Using A Radial Basis Function Neural Network*, NEUREL 2004, Belgrade, Proceedings, pp.73-78
[2] S.Haykin, *Neural Networks*, Mcmillan Publishing Co, 1994, Englewood Cliffs, pp.397-443
[3] Hecht-Nielsen, Neurocomputing, 1990, Addison-Wesley Publishing Company, pp.63-74
[4] S.C.Ahalt, A.K. Krishnamurty, P.Chen,D.E.Melton, *Competitive Algorithms for Vector Quantization*, (1990), Neural Networks, **3**, pp.277-291