Soft Competitive Learning in the Extended Differentiator Network

Seyed Jalal Kia & George Coghill

Electrical & Electronic Engineering Department
The University of Auckland
Auckland, New Zealand

Abstract

A two layer neural network called an Extended Differentiator Network (EDN) which combines unsupervised and supervised training is presented. The EDN uses a soft competitive learning method in the unsupervised layer followed by a supervised associative layer.

The soft competitive learning in the EDN takes the activity of all the competing neurons into account by using a one-step lateral inhibition mechanism. The functionality of the network is tested on a vowel recognition task and a cluster analysis problem. The simulation results indicate an effective use of the competing neurons resulting in a high recognition rate in a network with simple configuration.

1. Introduction

In a competitive learning system, a group of modules are used to produce the representative vectors of different clusters in the set of input vectors. This is a neural network implementation of the classical *K*-means clustering. In the clustering task, cluster centroids are modified in adaptation to the input data points to minimize an energy function which is the total average distortion error.

In the well known Winner-Take-All (WTA) type of competition, only the best matching or winning weight vector is adapted to become more similar to the input vector. The WTA method fails to utilize the competing neurons efficiently (underutilization problem) and is frequently confined in local minima of the energy function. Two common methods used for overcoming the above problems are adding a conscience to the system by incorporating the winning rate of the neurons into the updating formula [1] or using a soft-max adaptation rule that affects all cluster centres depending on their proximity to the input vector [2].

The authors of this paper have developed a Dynamic Competitive Learning (DCL) procedure [3,4] which is a form of soft competitive learning. In this paper, we propose a two layer neural network called an Extended Differentiator Network (EDN) to learn mapping operations and classify patterns. The EDN which is somewhat similar to the counterpropagation network of Hecht-Nielsen [5] combines unsupervised and supervised training. The unsupervised layer of the EDN uses our variation of soft competitive learning to find the clusters in the input feature vectors and the supervised layer associates the responses of the unsupervised layer to the desired vectors.

We have chosen the task of recognizing time-varying signals of 5 vowels and a two-dimensional cluster analysis problem to show the functionality of the EDN by way of computer simulations. In section 2, our variation of soft competitive learning is reviewed. In section 3, the EDN is presented. Section 4 gives the simulation results and section 5 concludes the paper.

2. Soft Competitive Learning

In soft competition, all the neurons are updated but the amount of update depends on how close the neuron's weight vector is to the current input vector. Nowlan [6] used this concept to model data distribution by overlapping radial basis functions in the context of maximum likelihood approach. In the maximum-entropy clustering technique suggested by Rose et. al. [7], the absolute distance between the weight vectors and the current input vector determines the adaptation step and an annealing process is used to reduce the temperature of the system.

Martinetz et. al. [2] have recently proposed a soft-max rule which is an extension of classical K-means clustering procedure and takes into account a neighbourhood-ranking within the input space. To find the neighbourhood-rank, each neuron has to compare its distance with the input vector to the distance of all the other neurons. This method has a high performance but is very computationally intensive. Kohonen's feature map algorithm [8] also uses a soft-max adaptation rule for the neurons which are in the neighbourhood of the winning neuron. Here, a neighbourhood-rank within an external lattice (instead of the input space) is considered.

In our DCL method, a novel type of neuron called the control neuron is incorporated into the system to implement a lateral inhibition mechanism and provide feedback reinforcement signals for updating the weights. The reinforcement signal combines the activity of all the competing neurons. Thus, in this method, all the competing units have their own share in the competition and all the weights impinging upon them are updated at each step of training. As a consequence of this kind of competition, the weight vectors of strongly responding units move faster towards an input pattern than the others. The adaptation rule is given by

$$\boldsymbol{w}_j(t+1) = S_j\,(\,\boldsymbol{x}(t) - \boldsymbol{w}_j(t)\,) + \boldsymbol{w}_j(t) \qquad (1)$$

0-7803-1901-X/94 $4.00 ©1994 IEEE

where $x(t)$ is the input vector, $w_j(t)$ is the weight vector of the jth competing neuron whose win-loss factor is represented by S_j. The formal presentation of the DCL method is given in the next section.

Based on the DCL algorithm, an unsupervised clustering network called a differentiator was proposed by the authors [4]. The weight vectors in the differentiator are utilized efficiently and they converge rapidly to the cluster centroids. The clustering results are also very much independent of the initial weight values. These effects can be seen in an example shown in Fig. 1. This example demonstrates the movement of 8 weight vectors, starting from random initial positions as shown by the labels, to the cluster centroids during two iterations of the data set. For a complete description of this example refer to [4].

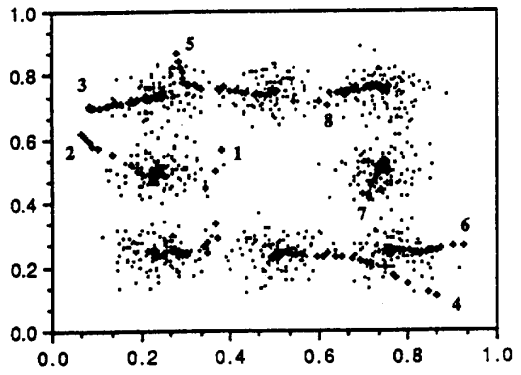


Fig. 1: Eight clusters of randomly generated Gaussian two-dimensional patterns and convergence of the weight vectors during two iterations of training in the differentiator network.

3. The Extended Differentiator Network

The EDN may be conceptually represented by a two layer network as shown in Fig. 2 accompanied by the implementation of the lateral inhibition mechanism shown in Fig. 3.

The network has three layers of neurons; input neurons, intermediate neurons (hidden neurons) and output neurons. The input neurons serve as fanout units and are connected to the neurons of the intermediate layer (hidden layer) through variable connection weights. Each input neuron is connected to every hidden neuron.

Associated with each hidden neuron is a control neuron to implement a lateral inhibition mechanism as shown in Fig. 3. The control neuron receives a fixed excitatory connection from the corresponding hidden neuron and fixed inhibitory connections from other hidden neurons as well as a *Bias* value. The output of the control neuron is used as a feedback in updating the weight vectors feeding the hidden layer.

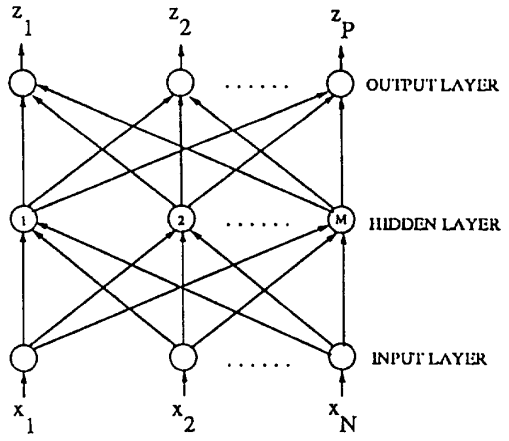


Fig. 2: Topology of the EDN.

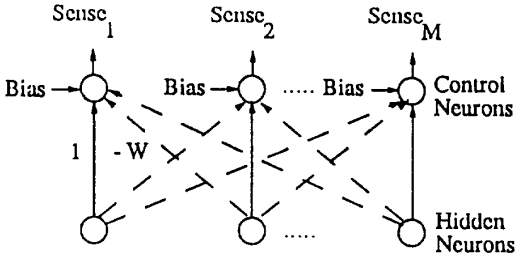


Fig. 3: Implementation of the lateral inhibition mechanism within the hidden layer.

The hidden neurons are also connected to the output neurons through another set of variable connection weights similar to the previous layer. These weights are updated by using the Grossberg outstar algorithm [9] which requires the desired vectors to be supplied at the output neurons by an external teacher. The weight vectors of this layer are updated in the second phase of training.

Formally, an n-dimensional input vector composed of real numbers is applied to the input layer. The weight vector of the hidden neuron j is denoted by w_j. The squared Euclidean distance d_j between weight vector w_j and the input vector x is:

$$d_j^2 = \sum_i (x_i - w_{ij})^2 \qquad (2)$$

where w_{ij} is the connection strength between the ith input neuron and the jth hidden neuron, and x_i is the ith element of the input vector x. To obtain the activation value y_j for the hidden neuron j, the calculated distance is passed through a nonlinear Gaussian function:

$$y_j = \exp(- d_j^2 / 2\rho^2) \quad (3)$$

The characteristic distance ρ is a selectivity parameter that determines the degree of fine tuning of the hidden neurons to the input vectors. In order to hold a dynamic competition, the activation values for all the hidden neurons are calculated and fed to the control neurons through fixed connection strengths. Each associated control neuron receives a fixed excitatory connection from its own hidden neuron with strength 1 and fixed inhibitory connections from other hidden neurons with strength W ($0 < W < 1$) as well as a *Bias* value. The output of the control neuron k called $Sense_k$ is calculated from equation (4), where g is a squashing function which confines the signal between 0 and 1.

$$Sense_k = g (y_k - W \sum_{j \neq k} y_j + Bias) \quad (4)$$

The learning rule used in the unsupervised layer of the network is based on a neurobiologically inspired process called sensitization. By sensitization we mean that a neuron is made to produce a stronger response to a stimulus. This process has roots in competitive learning [4]. Equation (5) represents the learning rule in which S is the sensitization constant which determines the rate of learning:

$$\Delta w_{ij} = S \, . \, Sense_j (x_i - w_{ij}) \quad (5)$$

At each step of training, the activation values for the hidden neurons are calculated using equations (2) and (3). Then the output of the control neurons are calculated from equation (4) and normalized. The next step in the training algorithm is to modify all the weight vectors according to equation (5). The above procedure is repeated for the number of times specified by the user.

Note that the *Sense* value used in equation (5) is obtained by performing only one step of the lateral inhibition mechanism. Hence, this method saves computation time and retains most of the information concerning the matching quality between the weight vectors and the input vector.

Having trained the first layer, the learning law for the change of w_{ij} is turned off. Then, by applying an input vector, a set of y_j values are obtained. To determine the vectors feeding the supervised layer the first k highly responding hidden neurons are chosen (k = 1 suffices for many cases) and their activities are normalized so that their sum equals 1. The activity of the other hidden neurons is set to 0.

The weight vectors of the second layer are updated using equation (6), where β is a factor which decreases with time as training proceeds, d_k is the k component of the desired vector and v_{jk} is the weight value between the hidden neuron j and the output neuron:

$$\Delta v_{jk} = \beta (d_k - v_{jk}) y_j \quad (6)$$

After training, the actual activation value for the output neuron k, denoted by z_k, is calculated from equation (7) where M is the number of hidden neurons:

$$z_k = \sum_{j=1}^{M} y_j \cdot v_{jk} \quad (7)$$

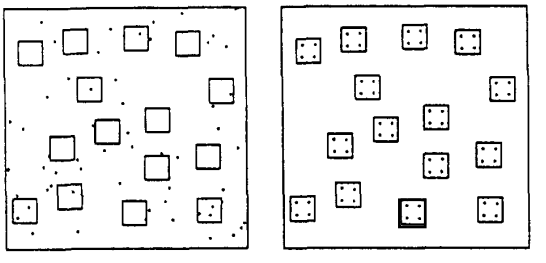
4. Simulation Results

In this section, we present two simulation examples. The first one demonstrates the clustering task of the unsupervised layer of the EDN, while the second example considers the complete EDN.

EXAMPLE 1:

In this example, we focus on the performance of the unsupervised layer of the EDN on a two-dimensional clustering problem described in Martinetz et. al. [2]. In this regard, the DCL algorithm is compared with K-means clustering, Kohonen-map and Maximum-entropy method mentioned in section 1.

The input data is drawn from 15 square shape clusters in a two-dimensional space as shown in Fig. 4.a. Also shown in this figure, are the randomly determined initial positions of 60 weight vectors. The global minimum for this clustering problem is known and it can be achieved if the weight vectors converge to the positions shown in Fig. 4.b.



(a) (b)

Fig. 4: A data distribution consisting of 15 separated clusters of square shape and positions of 60 weight vectors adapted from Martinetz et. al. [2].
(a) Weights have been initialized randomly.
(b) Weights have converged to a position which gives the global minimum of total average distortion error.

The performance of each algorithm can be measured from $(E - E_0) / E_0$ where E is the average distortion

error at time t and E_0 is the minimal distortion error.

Fig. 5 compares the performance of 4 algorithms in minimizing the distortion error with respect to the adaptation steps. Since the clustering result of the K-means algorithm is very much dependent on the initial values, the weights in this method have been initialized to the values taken from the given clusters; i.e., the a priori knowledge available has been used in favour of this algorithm. That is the main reason it converges quickly to its final distortion level.

The maximum-entropy method is capable of giving the lowest level of distortion. However, this is achieved after presenting a large number of patterns to the system and performing a high amount of computation for the annealing process. In this process, the temperature of the system must be carefully reduced to avoid the local minima.

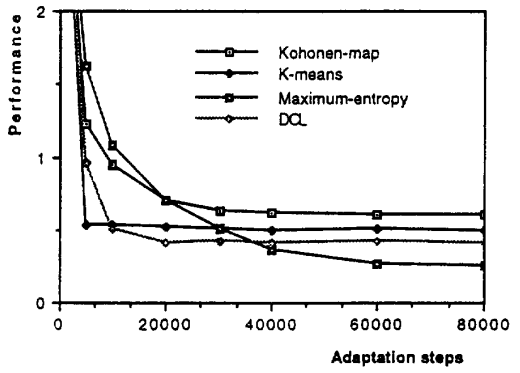


Fig. 5: The performance of 4 clustering algorithms in minimizing the distortion error with respect to the adaptation steps. The performance measure is given by $(E - E_0) / E_0$.

To show the behaviour of the DCL algorithm, only the unsupervised layer of the EDN was used. This layer consisted of 2 input neurons to represent the input data values and 60 competing neurons. In order to reduce the wandering of weights in the vicinity of the cluster centroids, the sensitization learning rate was decreased from an initial given value (S_{int}) as training proceeded.

The results shown for the DCL algorithm are the average of 10 runs of data with different initial positions of the weights. The parameters of the system were chosen as follows:

$Bias = 0.5$, $S_{int} = 0.01$, $W = 0.1$, $\rho = 0.2$.

$Bias$ and S_{int} were fixed to the given values, then W and ρ were found after a few trials.

The DCL algorithm performs better than the Kohonen-map and the K-means algorithm both in terms of speed and level of distortion. The DCL method is much faster than the maximum-entropy method and it quickly converges to the vicinity of the global minimum; then it starts to wander in that region. Since it does not perform the annealing process, the final positions of the weights will not be as close to the global minimum as those of the maximum-entropy method.

EXAMPLE 2:

In this example, we used the time varying signals of 5 vowels (/ɛ,i,ʌ,a,u/) uttered by a male speaker and sampled at 10 KHZ. In order to classify the signals, we extracted 12 Linear Prediction Coding (LPC) coefficients for each frame (200 samples) of the signals using the covariance method [10]. Successive frames overlapped by 100 samples. The 12 coefficients were normalized to lie between 0.0 and 1.0.

The input data file to the network consisted of a total of 250 12-dimensional vectors for the 5 vowels (50 vectors for each vowel). we corrupted the signals with 30% noise. A segment of the noisy vowel /ɛ/ is shown in Fig.6.

By initializing the 5 weight vectors of the first layer and using the network in a pattern matching style, a recognition rate of 88.8% was achieved which is very low. This result is summarized in Table 1.

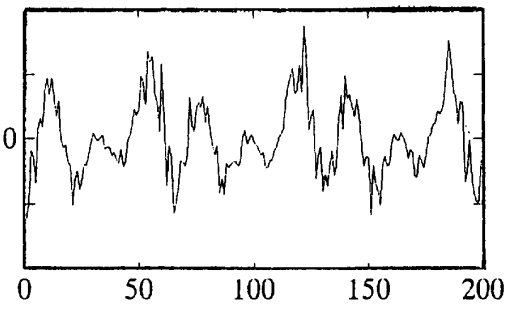


Fig. 6: Signal representing vowel /ɛ/ corrupted by 30% noise.

Network Size	12-5-5	12-10-5	12-20-5
Training Iterations (First Phase + Second Phase)	0 + 10	10 + 10	10 + 10
%Recognition Rate	88.8	93.6	98.8

Table 1: Classification results for the vowel recognition experiment for 3 diferent sizes of the network with input signals corrupted by 30% noise.

To increase the recognition rate we tried to use the network in a more effective manner by chosing a suitable configuration and training the first layer as well as the second layer. The results of two different cases are also shown in Table 1. The first 125 patterns were used for training and the remaining 125 patterns were used for testing the network. With 10 hidden neurons (using the first 2 vectors of each class for initializing the first layer weight vectors) and 10 iterations in the first phase of training, the recognition rate increased to 93.6%. With 20 hidden neurons (using the first 4 vectors of each class for initializing the first layer weight vectors) and 10 iterations in the first phase of training the recognition rate increased to as high as 98.8%.

The parameters of the network for the vowel recognition experiment were set to the following values:
$Bias = 0.3 \quad S = 0.05 \; W = 0.3 \quad \rho = 0.3.$

5. Conclusions

We presented a two-layer supervised network based on a soft variation of competitive learning. With our soft competitive learning method, all the neurons participate in the competition effectively and the weights converge quickly to the cluster centroids.

The simulation results for a vowel recognition task and a cluster analysis problem were presented. The results showed that the use of soft competitive learning can lead to high recognition rates without requiring long training sessions.

In our future research, we will concentrate on the analysis of the network to find methods for automatic selection of parameters and will apply the network to some real world problems.

References

[1] S. C. Ahalt, A. K. Krishnamurthy, P. Chen, and D. Melton, "Competitive learning algorithms for vector quantization," *Neural Networks*, vol. 3, no. 3, pp. 277-290, 1990.

[2] T. M. Martinetz, S. G. Berkovich, and K. S. Schulten, "Neural-Gas network for vector quantization and its application to time-Series prediction," *IEEE Trans. on Neural Networks*, vol. 4, no. 4, pp. 558-568, 1993.

[3] S. J. Kia and G. G. Coghill, "High performance clustering using dynamic competitive learning," *Electronics Letters*. vol. 28, no. 18, pp. 1753-1755, 1992.

[4] S. J. Kia, and G. G. Coghill, "Unsupervised clustering and centroid estimation using dynamic competitive learning," *Biological Cybern.* vol. 67, pp. 433-443, 1992.

[5] R. Hecht-Nielsen, "Applications of counterpropagation networks," *Neural Networks*, vol. 1, pp 131-139, 1988.

[6] S. J. Nowlan, "Maximum likelihood competitive learning," in *Advances in Neural Information Processing Systems* 2, D. Touretzky Ed., New York, Morgan Kauffman, 1990, pp. 574-582.

[7] K. Rose, F. Gurewitz, and G. Fox, "Statistical mechanics and phase transitions in clusterng," *Physicall Rev. Lett.*, vol. 65, no. 8, pp. 945-948, 1990.

[8] T. Kohonen, *Self-Organization and Associative Memory*. Springer-Verlag, 1989, third ed.

[9] S. J. Kia and G. G. Coghill, "A mapping neural network using unsupervised and supervised training," In *Proc. Int. Joint Conf. Neural Networks (IJCNN-91)*, Seattle, USA, July 1991, vol. 2, pp. 587-590.

[10] J. Makhul, "Linear prediction: A tutorial review," in *Proc. IEEE*, vol. 63(4), pp. 561-580, 1975.