Color Pattern Recognition on the Random Neural Network Model

Jose Aguilar and Valentina Rossell

CEMISID. Dpto. de Computación
Facultad de Ingeniería. Universidad de los Andes
Av. Tulio Febres. Mérida, 5010, Venezuela
aguilar@ing.ula.ve

Abstract. The purpose of this paper is to describe the use of the multiple classes random neural network model to learn various patterns having different colors. We propose a learning algorithm for the recognition of color patterns based upon the non-linear equations of the multiple classes random neural network model using gradient descent of a quadratic error function. Our model is defined for *nC* parameters for the whole network, where *C* is the number of colors, *n* is the number of pixels of the image, and each neuron is used to obtain the color value of each pixel in the bit map plane.

1. Introduction

The Random Neural Network (RNN) has been proposed by Gelenbe in 1989 [6,7,8]. This model calculates the probability of activation of the neurons in the network. Signals in this model take the form of impulses which mimic what is presently known of inter-neural signals in biophysical neural networks. The RNN has been used to solve optimization [1,2,3] and pattern recognition problems [4,5]. Fourneau and Gelenbe have proposed an extension of the RNN, Multiple Classes Random Neural Network (MCRNN) [9]. The problem addressed in this paper concerns the proposition of a learning algorithm for the recognition of color patterns, using MCRNN. We shall use each class to model a color. We present a backpropagation type learning algorithm for the MCRNN, using gradient descent of a quadratic error function when a set of input-output pairs is presented to the network. Thus, it requires the solution of a system of *nC* non-linear equations each time the n-neurons network learns a new input-output pair (n-pixels image with C colors). This work is organized as follows, in section 2 the theoretical bases of MCRN are reviewed. Section 3 presents our learning algorithm for MCRNN. In section 4, we present color pattern recognition applications. Remarks concerning future work and conclusions are provided in section 5.

2. The Multiple Classes Random Neural Model

The MCRNN is composed of *n* neurons and receives exogenous positive (excitatory) and negative (inhibitory) signals as well as endogenous signals exchanged by the neurons. Excitatory and inhibitory signals are sent by neurons when they fire, to other

R. Loganantharaj et al. (Eds.): IEA/AIE 2000, LNAI 1821, pp. 561-566, 2000.

neurons in the network or to outside world. In this model, positive signals may belong to several classes and the potential at a neuron is represented by the vector $\underline{K_i}=(K_{i1}, ..., K_{iC})$, where K_{ic} is the value of the "class *c* potential" of neuron *i*, or its "excitation level in terms of class *c* signals", and negative signals only belong to a single class. The total potential of neuron *i* is $K_i = \Sigma_{c=1}^{C} K_{ic}$. When a positive signal of class *c* arrives at a neuron, it merely increases K_{ic} by 1, and when a negative signals arrives at it, if Ki>0, the potential is reduced by 1, and the class of the potential to be reduced is chosen randomly with probability Kic/Ki for any c=1, ..., C. Exogenous positive signals of class *c* arrive at neuron *i* in a Poisson stream of rate $\Lambda(i, c)$, while exogenous negative signals arrive at it according to a Poisson process of rate $\lambda(i)$. A neuron is excited if its potential is positive (Ki>0). It then fires, at exponentially distributed intervals, sends excitatory signals of different classes, or inhibitory signals, to other neurons or to the outside of the network. The neuron *i* sends excitatory signals of class *c* at rate r(i, c)>0, with probability Kic/Ki. When the neuron fires at rate r(i, c), deletes by 1 its class *c* potential and sends to neuron *j* a class φ positive signal with probability $p^+(i, c; j, \varphi)$, or a negative signal with probability $p^-(i, c; j)$. On the other hand, the probability that the deleted signal is sent out of the network is d(i, c). Let q(i, c) with $0 < q(i, c) < 1$ be the solution of the system of non-linear equations:

$$q(i,c) = \lambda^+(i, c)/(r(i, c)+\lambda^-(i)) \qquad (1)$$

where, $\lambda^+(i, c) = \Sigma_{(j, \varphi)} q(j, \varphi)r(j, \varphi)p^+(j, \varphi ; i, c)+\Lambda(i, c)$

$\lambda^-(i) = \Sigma_{(j, \varphi)} q(j, \varphi)r(j, \varphi)p^-(j, \varphi; i)+\lambda(i)$

The synaptic weights for positive ($w^+(j, \varphi; i, c)$) and negative ($w^-(j, \varphi; i)$) signals are defined as:

$$w^+(j, \varphi; i, c) = r(j, \varphi)p^+(j, \varphi; i, c) \qquad w^-(j, \varphi; i) = r(j, \varphi)p^-(j, \varphi; i)$$

and, $r(j, \varphi) = [\Sigma_{(i, c)} w^+(j, \varphi; i, c) + \Sigma_{(i, c)} w^-(j, \varphi; i)]$

3. Learning Algorithm on the Multiple Classes Random Neural Network Model

We propose a gradient descent algorithm for choosing the set of network parameters $w^+(j, z; i, c)$ and $w^-(j, z; i)$ in order to learn a given set of *m* input-output pairs (X, Y) where the set of successive inputs is denoted by:

$$X = \{X_1, ..., X_m\} \qquad X_k= \{X_k (1,1), ..., X_k (n, C)\}$$

where, $X_k (i, c)$ is the c^{th} class on the neuron *i* for the patron *k*

$$X_k(i, c) = \{\Lambda_k(i, c),\ \lambda_k(i)\}$$

and the successive desired outputs are the vector

$$Y = \{Y_1, ..., Y_m\}$$

where, $Y_k = \{Y_k(1,1), ..., Y_k(n, C)\}$, and $Y_k(1,1)=\{0, 0.5, 1\}$

The values $\Lambda_k(i, c)$ and $\lambda_k(i)$ provide the network stability. Particularly, in our model $\Lambda(i, c)=Lic$ and $\lambda(i)=0$, where Lic is a constant for the class c of the neuron i. $X_k(i, c)$ are initialized as follows:

$$Y_k(i, c)>0 \Rightarrow X_k(i, c) = (\Lambda_k(i, c), \lambda_k(i)) = (Lic, 0)$$
$$Y_{ik}(i, c)=0 \Rightarrow X_k(i, c) = (\Lambda_k(i, c), \lambda_k(i)) = (0, 0)$$

The rule to update the weights may be written as:

$$w_k^+(u,p; v,z) = w_{k-1}^+(u,p; v,c) - \mu\Sigma^n_{i=1}\Sigma^C_{c=1}(q_k(i,c) - y_k(i,c))[\delta q(i,c) / \delta w^+(u,p; v,z)]_k$$

(2)

$$w_k^-(u,p; v) = w_{k-1}^-(u,p; v) - \mu\Sigma^n_{i=1}\Sigma^C_{c=1}(q_k(i,c) - y_k(i,c))[\delta q(i,c) / \delta w^-(u,p; v)]_k$$

where, $\mu > 0$ is the learning rate (some constant).

$q_k(i)$ is calculated using X_k, $w^+_k(u, p; v, z) = w^+_{k-1}(u, p; v, z)$ and

$w^-_k(u, p; v) = w^-_{k-1}(u, p; v)$ in (1)

$[\delta q(i,c) / \delta w^+(u,p;v,z)]_k$ and $[\delta q(i,c) / \delta w^-(u,p;v)]_k$ are evaluated using the

values $q(i,c) = q_k(i,c)$, $w^+_k(u, p; v, z) = w^+_{k-1}(u, p; v, z)$ and

$w^-_k(u, p; v) = w^-_{k-1}(u, p; v)$ in (2)

and, $\delta q(i, c)/ \delta W^+(u,p;v,z) = \gamma^+(u,p;v,z)/q(u,p)\ [I-W]^{-1}$

$\delta q(i, c)/ \delta W^-(u,p;v) = \gamma^-(u,p;v)/q(u,p)\ [I-W]^{-1}$

if (u=i) and (v≠i) then

$\gamma^+(u,p;v,z) = -1/D(i,c)$

$\gamma^-(u,p;v) = -1/D(i,c)$

if (u≠i) and (v=i) then

$\gamma^+(u,p;v,z) = 1/D(i,c)$

$\gamma^-(u,p;v) = -q(i,c)/D(i,c)$

if (u=i) and (v=i) then

$\gamma^+(u,p;v,z) = 0$

$\gamma^-(u,p;v) = -(1+ q(i,c))/D(i,c)$

if (u≠i) and (v≠i) then

$\gamma^+(u,p;v,z) = 0$

$\gamma^-(u,p;v) = 0$

finally, $D(i,c) = r(i, c)+ \Sigma^{n}_{j=1} \Sigma^{C}_{z=1} q(j, z) w^{-}(j, z; i)]$

$W = \Sigma^{n}_{j=1} \Sigma^{C}_{z=1} [w^{+}(j, z; i, c) + w^{-}(j, z; i)q(j,z)]/D(j,z)$

The complete learning algorithm for the network is:

- *Initiate the matrices W_0^+ and W_0^- in some appropriate manner. Choose a value of μ in (2).*
- *For each successive value of m:*
  - *Set the input-output pair (X_k, Y_k)*
  - *Repeat*
    - *Solve the equation (1) with these values*
    - *Using (2) and the previous results update the matrices W_k^+ and W_k^-*

Until the change in the new values of the weights is smaller than some predetermined valued.

4. Color Pattern Recognition Problem on the Multiple Classes Random Neural Network Model

4.1 Problem Definition

We now show how the MCRNN can be used to solve the Color Pattern Recognition problem. In this section, we present several examples to compare the quality of our learning algorithm for different pattern types. In our approach, a "signal class" represents obviously each color. To design such a memory, we have used a single-layer MCRNN of *n* fully interconnected neurons. For every neuron *i* the probability that emitting signals depart from the network is d(i, c)=0. We suppose a pattern composes by *n* points (j, k) in the plane (for j= 1, ..., J and k=1, ..., K). We associate a neuron N(i) to each point (j, k) in the plane (for i=1, ..., n; j= 1, ..., J and k=1, ..., K). The state of N(i) can be interpreted as the color intensity value of the pixel (j, k). That is, each pixel is represented by a neuron. In another hand, we suppose three classes to represent the primary colors (red, green, and blue) according to the RGB model. This model allows create different colors with the combination of different intensities of the primary colors. For example, to represent a pixel with red color the neuron value is (1, 0, 0), the black color is (1, 1, 1), the pink color is (0.5, 0, 0), etc. We suppose values equal to 0, 0.5 and 1 for each class on every neuron. In this way, we can represent geometric figures with different combinations of colors. The parameters of the neural network will be chosen as follows: $p^{+}(j, \varphi ; i, c) = p^{+}(i, c; j, \varphi)$ and $p^{-}(i, c; j) = p^{-}(j, c; i)$, $\forall i,j=1, ..., n$ $c,\varphi=1,..., C$. We will input various geometric figures to MCRNN and train the network to recognize these as separate categories. To evaluate our learning algorithm, we use a set of figures (group A) composes by the figures shown in figure 1, where blackened boxes represent blue colors, gray boxes represent green colors and white boxes represent red colors. Each figure is represented by a 6*6 grid of pixels. Thus, we use a single-layer MCRNN composed by 36 neurons (n=36) and 3 classes (C=3).

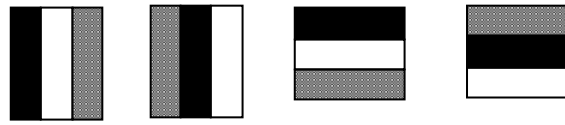

Fig. 1. Geometric Figures with three colors.

4.2 Results Analysis

The results for the first group are presented on figure 2. To evaluate the performance of the learning algorithms, we show the minimal errors reached during the learning phase and their execution times. These values represent the average of 8 processes for each set S_i of images. This algorithm provides a good error convergence for the learning phase. Particularly, the learning of the sets S_4 and S_6 remain good for our learning algorithm. Concerning S_2 and S_8, error costs increases.

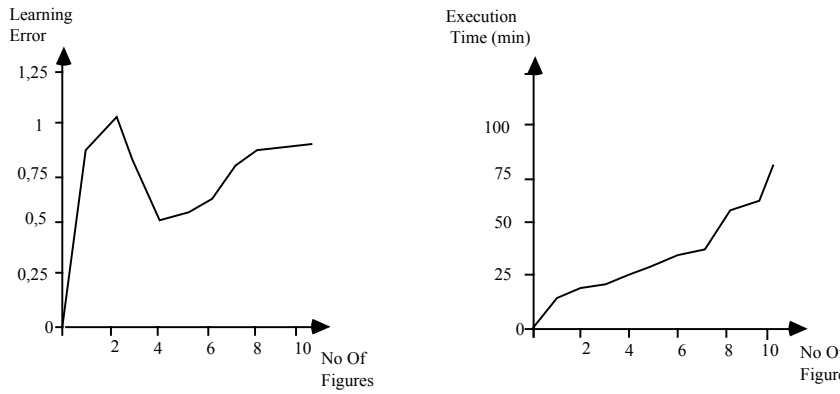

Fig. 2. Learning error and execution time of the learning algorithms

In order to test associative memories, we have evaluated the recognition rates of distorted versions of the training patterns using the recognition algorithm proposed in [4]. These values represent the average of 8 processes for each set S_i of images. We generated *20* noisy images used as inputs, for each training image and for a given distortion rate. The result of the learning stage is used as the initial neural network of this second phase (retrieval stage). We have corrupted them by reasonable noise rates equal to *0%, 15%* and *30%* distortion by modifying bit values at random. A pattern is recognized if the residual error rate is less than *3*. The results are presented on figure 3. The performance results obtained are lower when the noise rate is important (memories are then more discriminating). Our algorithm provides a good recognition rate. Particularly, the recognition rate of the sets S_4 and S_6 remain good for our approach. Concerning S_{10} and 30% of noise rate, recognition rate decreases.

Noisy Rate	0%			15%			30%		
Number of Figures	4	6	10	4	6	10	4	6	10
Recognition Rates	99%	99%	97%	88%	87%	84%	73%	71%	67%

Fig. 3. Recognition rate of noisy versions of figures.

5. Conclusions

In this paper, we have propose a learning algorithm based on the Multiple Classes Random Neural Model. We have shown that this model can efficiently work as associative memory. We can learn arbitrary colour images with this algorithm, but the processing time will increase rapidly according to the number of pixels and colours used. The number of neurons is dictated by the image resolution (in our case, we are test for 6*6 pixels). During the learning phase, we have met classical problems like the existence of local minima and large learning times. However, most of the computations are intrinsically parallel and can be implemented on SIMD or MIMD architectures. Next work will study a new retrieval algorithm adapted to these types of figures.

References

1. Aguilar, J.: Evolutionary Learning on Recurrent Random Neural Network. Proc. of the World Congress on Neural Networks, International Neural Network Society (1995) 232-236.
2. Aguilar, J.: An Energy Function for the Random Neural Networks. Neural Processing Letters **4** (1996) 17-27.
3. Aguilar, J.: Definition of an Energy Function for the Random Neural to solve Optimization Problems. Neural Networks **11** (1998) 731-738.
4. Aguilar, J., Colmenares A.: Resolution of Pattern Recognition Problems using a Hybrid Genetic/Random Neural Network Learning Algorithm. Pattern Analysis and Applications **1** (1998) 52-61.
5. Atalay, V., Gelenbe, E., Yalabik, N.: The random neural network model for texture generation. Intl. Journal of Pattern Recognition and Artificial Intelligence **6** (1992) 131-141.
6. Gelenbe, E.: Random neural networks with positive and negative signals and product form solution. Neural Computation **1** (1989) 502-511.
7. Gelenbe, E.: Stability of the random neural networks. Neural Computation **2** (1990) 239-247.
8. Gelenbe, E.: Learning in the recurrent random neural network. Neural Computation **5** (1993) 325-333.
9. Fourneau, M., Gelenbe, E., Suros, R.: G-networks with Multiple classes of negative and positive customers. Theoretical Computer Science, **155** (1996) 141-156.