Rev. Téc. Ing. Univ. Zulia. Vol. 28, Nº 1, 51 - 58, 2005

The random neural network model on the color pattern recognition problem

José Aguilar

CEMISID. Departamento. de Computación. Facultad de Ingeniería. Universidad de Los Andes. Mérida-Venezuela. E-mail: aguilar@ing.ula.ve. Phone (58.74) 2402914; Fax (58.74) 2402872

Abstract

The purpose of this paper is to describe the use of the multiple classes random neural network model to recognize 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. In addition, we propose a progressive retrieval process with adaptive threshold value. The experimental evaluation shows that our approach provides good results.

Key words: Multiple classes random neural network, color pattern recognition, learning algorithm, retrieval process.

El modelo de redes de neuronas aleatorias en el problema de reconocimiento de patrones coloreados

Resumen

El propósito de este artículo es describir el uso del modelo neuronal aleatorio con múltiples clases para reconocer patrones con diferentes colores. Nosotros proponemos un algoritmo de aprendizaje para el reconocimiento de patrones coloreados basados en la ecuación no lineal del modelo neuronal aleatorio con múltiples clases usando el descenso de gradiente de una función cuadrática de error. Además, proponemos un proceso de recuperación progresiva con un valor de umbral adaptativo. La evaluación experimental muestra que nuestro enfoque da buenos resultados.

Palabras clave: Modelo neuronal aleatorio con múltiples clases, reconocimiento de patrones coloreados, algoritmo de aprendizaje, proceso de recuperación.

1. Introduction

Humans use color, shape and texture to understand and recollect the contents of a pattern. Therefore, it is natural to use features based on these attributes for pattern recognition [8, 9, 16, 17]. The effectiveness of using simple color features for pattern recognition is demonstrated in [15]. Colombo et al. described a system for pictorial content representation and recognition based on color distribution features [8] in which the distribution of chromatic content in a pattern is described through a set of color histograms and a pattern matching strategy using this set. Mojsilovic et al. recently determined the basic categories (vocabulary) used by humans in judging similarity of color patterns, their relative importance and relationsh ips, as well as the hierarchy of rules (grammar) [17].

In this paper a color pattern recognition approach composed by a learning algorithm and a retrieval procedure for the multiple classes RNN is proposed. We shall use each class to model a color. We present a backpropagation type learning algorithm for the recurrent multiple classes RNN model using gradient descent minimization of a quadratic error function when a set of input-output pairs is presented to the network. Our model is defined for nC parameters for the whole

network, where C is the number of primary 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. The primary colors create different colors according to the RGB model. Thus, our learning algorithm 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 primary colors). In addition, we propose a progressive retrieval process with adaptive threshold value.

The Random Neural Network (RNN) was proposed by Gelenbe in 1989 [11, 12, 13]. This model does not use a dynamic equation, but uses a scheme of interaction among neurons. It calculates the probability of activation of the neurons in the network. Signals in this model take the form of impulses that mimic what is presently known as inter-neural signals in biophysical neural networks. The RNN has been used to solve optimization [1, 2, 4] and pattern recognition problems [3, 5, 7]. Gelenbe considered a learning algorithm for recurrent random neural network model [14], and we have proposed modifications of this algorithm for combinatorial optimization problems [4] and an evolutionary learning for combinatorial optimization and recognition problems [1, 5]. Fourneau et al. have proposed an extension of the RNN, called multiple classes random neural network model [10].

This work is organized as follows, in section 2 we present the multiple classes RNN. Section 3 presents our recognition algorithm (learning and retrieval processes) for multiple classes RNN. In section 4, we present some applications. Remarks concerning future work and conclusions are provided in section 5.

2. The Multiple Classes Random Network Model

The neural network is composed of n neurons and receives exogenous positive (excitatory) and negative (inhibitory) signals as well as endogenous signals exchanged by the neurons. As in the classical model [11, 12, 13], neurons send excitatory and inhibitory signals when they fire, to other 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 $\underline{K}_i=(K_{i1},\ldots, K_{iC})$, where K_{ic} is the value of the "class c potential" of neuron i, or its "excitation level in terms of class c signals"; negative signals only belong to a single class. The total potential of neuron i is $K_i=\sum_{c=1}^{C} K_{ic}$. The arrival of an excitatory signal of some class increases the corresponding potential of a neuron by 1, while an inhibitory signal's arrival decreases it by 1. That is, when a positive signal of class c arrives at a neuron, it merely increases K_{ic} by 1, while when a negative signal arrives at it and if $K_i>0$, the potential is reduced by 1, with the class of the potential to be reduced chosen randomly with probability K_{ic}/K_i for any $c=1,\ldots, C$. A negative signal arriving at a neuron whose potential is zero has no effect on its potential.

Exogenous positive signals of class c arrive at neuron i at 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. It then fires at exponentially distributed intervals, sending excitatory signals of different classes, or inhibitory signals, to other neurons or to the outside of the network. That is, neuron i can fire when its potential is positive ($K_i>0$) and sends excitatory signals of class c at rate $r(i, c)>0$, with probability K_{ic}/K_i. When neuron i fires at rate $r(i, c)$ it 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, or that it is "lost", is $d(i, c)$. Clearly we shall have:

$$\sum_{(j,\varphi)} p^+(i,c;j,\varphi) + \sum_{j} p^-(i,c;j) + d(i,c) = 1$$

for $\forall i = 1, n$ and $c = 1, C$

Let $K(t)$ be the vector representing the state of the neural network at time t and $K=(K_1,\ldots, K_n)$ be a particular value of the vector. We shall denote by $p(K, t) = Pr[K(t)=K]$ the probability distribution of the state. The main property of this model is the excitation probability of the "class φ potential" of neuron j, $q(j,\varphi)$, which $0 < q(j,\varphi) < 1$ satisfies the non-linear equation [10]:

$$q(j,\varphi) = \lambda^+(j,\varphi) / \left(r(j,\varphi)+\lambda^-(j)\right) \qquad (1)$$

where,

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

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

Thus, p(K, t), the stationary probability distribution of network state, satisfies

$$p(K,t) = \prod_{i=i}^{n} \prod_{c=i}^{C} [1 - q(i,c)] q(i,c)^{k_{ic}} \qquad (2)$$

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

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

$w^{-}(i, c; j) = r(i, c)p^{-}(i, c; j)$

and, if d(i, c)=0, the fire rate r(i, c) will be

$$r(i, c) = [\sum_{(j,\varphi)} w^{+}(i, c; j, \varphi) + \Sigma_j w^{-}(i, c; j)] \qquad (3)$$

3. Color pattern recognition algorithm on the multiple classes random neural network model

We now show how the multiple classes RNN can be used to solve the Color Pattern Recognition problem, based on an associative memory technique [3, 5]. In our approach, a "signal class" represents obviously each color. To design such a memory, we have used a single-layer RNN 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 composed by n pixels (m, k) in the plane (for m= 1,..., J and k=1,..., K), each of which has associated a neuron N(i) (for i=1,..., n). The state of neuron N(i) can be interpreted as the color intensity value of the corresponding pixel. On the other hand, we suppose three classes to represent the primary colors (red, green, and blue) according to the RGB model. This model allows to 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 possible values of 0, 0.5 and 1 for each class on every neuron. In this way, we can represent geometric Figures with different combinations of colors. We have used this model because it agrees better with human chromatic perception [8], but the proposed approach can use another model like this one to represent the colors of a given pattern. The parameters of the neural network will be chosen as follows:

a) $p^{+}(j, \varphi ; i, c) = p^{+}(i, c; j, \varphi)\, p^{-}(i, c; j) = p^{-}(j, c; i)$ for any i,j=1,..., n and c,φ=1,..., C.

b) $\Lambda(i, c)$=Lic and $\lambda(i)$=0, where Lic is a constant for the class c of the neuron i.

The values in (a) were chosen because the neural network has a symmetric relationship between the neurons to guarantee the associative memory behavior of our approach. On the other hand, the values in (b) were chosen since an exogenous signal is sufficient to guarantee the network stability, and the value of Lic must be chosen accordingly (Equation 2).

3.1. Learning Algorithm

Now, we define a learning algorithm for the multiple classes RNN model. We propose a gradient descent algorithm for adjusting 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:

$X = \{X_1, ..., X_m\}$

where, $X_k = \{X_k(1,1), ..., X_k(n, C)\}$, and $X_k(i, c)$ is the c^{th} class on the neuron i for the k^{th} pair

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

and the successive desired outputs are

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

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 $\Lambda_k(i, c)$ and $\lambda_k(i)$ are initialized as have been defined previously. Typically, arrival rates of exogenous signals are chosen as follows:

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

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

The network approximates the set of desired output vectors such that the cost function E_k:

$$E_k = 1/2 \sum_{i=1}^{n} \sum_{c=1}^{C} [q_k(i,c) - Y_k(i,c)]^2 \qquad (4)$$

is minimized. The rule to update the weights may be written as:

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

$w_k^-(u,p;v) = w_{k-1}^-(u,p;v) - \mu\Sigma^n_{i=1}\Sigma^C_{c=1}(q_k(i,c) - y_k(i,c))[\delta q(i,c)/\delta w^-(u,p;v)]_k$ (5)

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)

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:
  - Select the input-output pair (X_k, Y_k)
  - Repeat
    - Solve equation (1) with these values
    - Using (5) 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.

For more details about this learning algorithm, see [6].

3.2. Retrieval Procedure

Once the learning phase is completed, the network must perform as well as possible the completion of noisy versions of the training vectors. In this case, we propose a progressive retrieval process with adaptive threshold value. Let X'= {X'(1, 1),..., X'(n, C)} be any input vector in which each X'(i, c) (i=1,..., n and c=1,..., C), may take values 0, 0.5 or 1. In order to determine the corresponding output vector Y= {Y(1,1),..., Y(n, C)} we first compute the vector of probabilities Q=(q(1, 1),..., q(n, C)). We consider the q(i, c) values such that *1-T<q(i, c)<T/2* or *1-T/2<q(i, c)<T*, with for instance T=0.8, belong to the uncertainty interval Z. When the network stabilizes to an attractor state, the number NB_Z of neurons whose q(i, c)∈Z is equal to 0. Hence, we first treat the neurons whose state is considered certain to obtain the output vector $Y^{(1)} = (Y^{(1)}(1,1),..., Y^{(1)}(n,C))$, with:

$$Y^{(1)}(i,c) = Fz(q(i,c)) = \begin{cases} 1 \text{ if } q(i,c) > T \\ 0 \text{ if } q(i,c) < 1-T \\ 0.5 \text{ if } T/2 \le q(i,c) \le 1-T/2 \\ x'_i \text{ otherwise} \end{cases}$$

where Fz is the thresholding function by intervals. When q(i, c) > T we can guarantee than q(i, c) =1, when T/2 ≤ q(i, c) ≤1-T/2 we can guarantee than q(i, c)=0.5, and when q(i, c) < 1-T we can guarantee than q(i, c)=0. If NB_Z = 0 this phase is terminated and the output vector is $Y=Y^{(1)}$. Otherwise, Y is obtained after applying the thresholding function f_β as follows:

$$Y(i,c) = f_\beta(q(i,c)) = \begin{cases} 1 \text{ if } q(i,c) > \beta \\ 0.5 \text{ if } \beta/2 < q(i,c) < \beta \\ 0 \text{ otherwise} \end{cases}$$

where β is the selected threshold. Each value q(i, c)∈Z is considered as potential thresholds. That is, for each $q(i,c) \in Z$:

$$\beta = \begin{cases} q(i,c) \text{ if } q(i,c) > 0.666 \\ 1 - q(i,c) \text{ otherwise} \end{cases}$$

Eventually, Z can be reduced by decreasing T (for T>0.666). For each potential value of β, we present the vector $X'^{(1)}(\beta) = f_\beta(Q)$ to the network. Then, we compute the new vector of probabilities $Q^{(1)}(\beta)$ and the output vector $Y^{(2)}(\beta) = Fz(Q^{(1)}(\beta))$. We keep the cases where NB_Z=0 and $X'^{(1)}(\beta) = Y^{(2)}(\beta)$. If these two conditions are never satisfied, the initial X' is considered too different of any training vector. If several thresholds are candidate, we choose the one which provides the minimal error (difference between q(i, c) and Y(i, c), for i=1, n and c=1,..., C):

$$E(\beta) = 1/2\ \Sigma^n_{i=1}[q(i,c)^{(1)}(\beta) - Y(i,c)^{(1)}(\alpha)]^2 \quad (6)$$

4. Experimental Results

4.1. Description of the examples

In this section we present several examples to evaluate the quality of our recognition algorithm for different pattern types. We will give various geometric Figures as inputs to a Multiple Classes Random Neural Network and

train the network to recognize them. To evaluate our approach, we use three Figure groups: the first group (group A) includes the set of Figures shown in Figure 1, where blackened boxes represent blue colors, gray boxes represent green colors and white boxes represent red colors. The next group (Group B), which is composed of the black and white Figures shown in Figure 2, is used to compare our approach with the recognition algorithm based on RNN proposed in [3], and the evolutionary learning approach proposed in [5]. The last group (group C) is composed by the set of patterns used in [17] (Figure 3). For group C extended experiments are presented to evaluate the performance of our method according to the relationship between the problem features (number of patterns, pixels and colors), recognition rates and processing time.

For the first and second cases, each Figure is represented by a 6*6 grid of pixels. For example, the seventh pattern in Figure 2 can be represented as shown in Figure 4. According to the RGB model, the black boxes are represented as (1, 1, 1), while white boxes are represented as (0, 0, 0). Hence, we can represent geometric Figures with different combinations of colors (for example, in Figure 2, if we suppose black boxes correspond to red colors, and white boxes to blue colors, neurons for black boxes are equal to (1, 0, 0) and for white boxes are equal to (0, 0, 1)). Thus, for these cases we use a single-layer multiple classes RNN composed by 36 neurons (n=36) and 3 classes (C=3).

4.2. Analysis of Results

In order to test associative memories, we have evaluated the recognition rates of distorted versions of the training patterns (Tables 1 and 2).

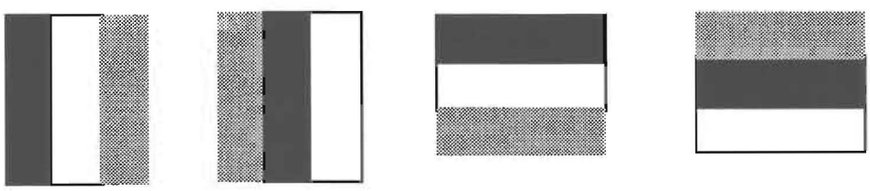


Figure 1. Geometric Figures with three colors (Group A).

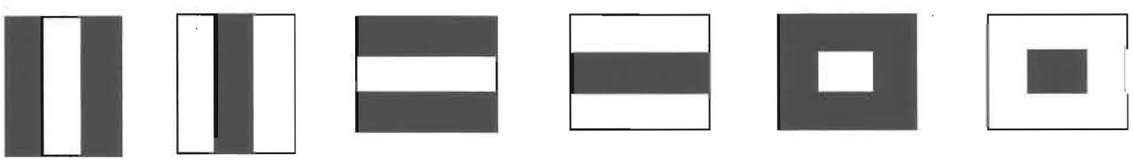


Figure 2. Geometric Figures with two colors (Group B)

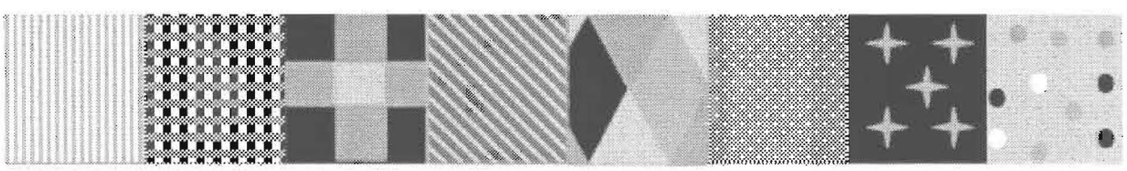
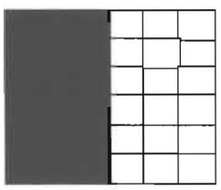


Figure 3. Pattern set used in the last experiment (Group C).



=>

111 111 111 000 000 000
111 111 111 000 000 000
111 111 111 000 000 000
111 111 111 000 000 000
111 111 111 000 000 000
111 111 111 000 000 000

Figure 4. Representation of a geometric Figure with a 6*6 pattern.

Table 1
Recognition rate of noisy versions of Group A

Noisy Rate	0%	10%	20%	30%								
Number of Figures	4	6	10	4	6	10	4	6	10	4	6	10
Group A	99%	99%	97%	94%	93%	89%	83%	82%	81%	73%	71%	67%

Table 2
Recognition rate of noisy versions of Group B

Noisy Rate	0%	10%	20%	30%								
Number of Figures	4	6	10	4	6	10	4	6	10	4	6	10
Cl	99%	95%	96%	93%	91%	85%	83%	80%	77%	68%	66%	62%
Evol	99%	99%	99%	96%	95%	92%	87%	85%	81%	75%	74%	72%
Mult	99%	99%	99%	95%	94%	90%	85%	84%	81%	73%	72%	70%

We generated *20* noisy versions of each training image, for a given distortion rate. The result of the learning stage is used as the initial neural network for this second stage (retrieval stage). We have corrupted them with noise rates of *10%*, *20%* 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 in tables 1 and 2. These values represent the average of 8 processes for each set S_i of images. The performance degrades when the noise rate is important (memories are then more discriminating). The results for the first group are presented on Table 1. Our algorithm provides a good recognition rate because we recognize a large number of patterns despite corrupted parts of the Figures (even when the noise rate is large (30%)). 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.

Table 2 shows the recognition rate for the last group of images (Group B) using the classical gradient recognition algorithm (*Cl*), the hybrid Genetic/Random Neural Network learning algorithm (*Evol*) and our Multiple Classes learning algorithm (*Mult*). In general, *Evol* appears to give the best results. The recognition rate remains good for our algorithm (*Mult*) if we compare its results with the results corresponding to *Evol*. It provides a better recognition rate that *Cl*, which is the algorithm with the worst performance.

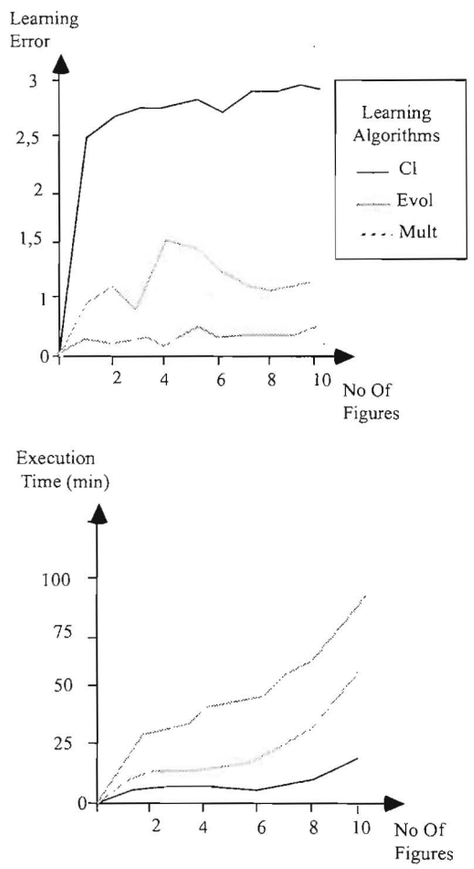


Figure 5. Learning error and execution time of the learning algorithms for Group B.

Table 3
Performance Evaluation of our method with the noisy versions of Group C

Noise Rate	0%				10%				20%			
Figures	1-8	1-3	6-8	1-5	1-8	1-3	6-8	1-5	1-8	1-3	6-8	1-5
Number of pixels	144	144	144	144	144	144	144	144	144	144	144	144
Recognition Rates	95%	98%	98%	94%	88%	92%	95%	90%	78%	83%	84%	80%
Retrieval Time (sec)	240	31	33	120	300	34	43	60	234	33	38	55
Number of pixels	576	576	576	576	576	576	576	576	576	576	576	576
Recognition Rates	95%	99%	99%	95%	89%	93%	96%	91%	79%	85%	86%	82%
Retrieval Time	720	120	123	212	723	114	132	221	756	134	144	321
Number of pixels	2304	2304	2304	2304	2304	2304	2304	2304	2304	2304	2304	2304
Recognition Rates	99%	99%	99%	99%	95%	97%	97%	96%	92%	95%	95%	92%
Retrieval Time	9188	1302	1287	2341	8923	1100	1098	2178	9012	1231	1101	2111

Figure 5 shows the system errors during the learning phase for Group B using the classical gradient descent learning algorithm (*Cl*), the hybrid Genetic/Random Neural Network learning algorithm (*Evol*) and our Multiple Classes learning algorithm (*Mult*). *Evol* gives the best results, but with a substantially large execution time due to a very slow convergence. The learning remains good for our learning algorithm (*Mult*) because the error is smaller than 1.5; with this error our approach can recognize a large number of Figures in the recognition stage. It provides a better error convergence on the learning phase than *Cl*. Regarding *Cl* its error is important, which justifies its bad recognition rate.

Table 3 shows the relationship between the problem features (number of pixels and colors), recognition rates and processing time for the set of Figures of the group C. If we increase the number of pixels to describe a pattern we improve the quality of the retrieval phase, but the execution time increases exponentially. The number of colors is not important because our system doesn't depend on it. If the RGB model can be used to represent a specific color of a given pattern, our approach can recognize it (see the similarity of the recognition rates for the cases of Figures 1-3 and 6-8). When the patterns are different (patterns 6, 7 and 8 in Figure 3) the system has a better retrieval rate. Our approach can recognize several patterns, but with a large retrieval time if we like to obtain good retrieval rates.

Our system has the typical drawback of an associative memory approach: low storage capacity. If we compare our approach with the method proposed on [8, 17], their approach has a better storage capacity (they tested their approach for 30 patterns), but our recognition rate is better ($\geq$ 90% for 20% of noise rate).

5. Conclusions

In this paper we have proposed a recognition algorithm based on the Multiple Classes Random Neural Model. We have considered Figures with several complexities, particularly the group C. We have tested the capacity of our model to recognize Figures with arbitrary number of colors, noise rate and number of pixels. We have shown that this model can efficiently work as an associative memory, and that we can recognize arbitrary color images, but the processing time will increase rapidly according to the number of pixels used. The number of neurons is given by the image resolution, which has a direct influence on the performance of our approach. During the learning phase we have found classical problems like the existence of local minimal and slow learning. Regarding the retrieval

algorithm, we have obained good performance but with a large execution time. However, most of the computations are intrinsically parallel and can be implemented on SIMD or MIMD architectures. In fact, we are currently working in a parallel version of our approach. We are going to extend our approach for geometric Figures where the colors have not a good definition, using the fuzzy logic theory.

Acknowledgment

This work was partially supported by CONICIT grant AP-97003817, CDCHT-ULA grant I-503-95-A05 and CeCalCULA (High Performance Computing Center of Venezuela).

References

1. Aguilar J. Evolutionary Learning on Recurrent Random Neural Network, In Proceeding of the World Congress on Neural Networks, 1995, pp. 232-236.
2. Aguilar J. An Energy Function for the Random Neural Networks, Neural Processing Letters, 1996, 4: 17-27.
3. Aguilar J. A Recognition Algorithm using the Random Neural Network, In Proceeding. of the 3rd. International Congress on Computer Science Research, 1996, pp. 15-22.
4. Aguilar J. Definition of an Energy Function for the Random Neural to solve Optimization Problems, Neural Networks, 1998, 11: 731-738.
5. Aguilar J., Colmenares A. Resolution of Pattern Recognition Problems using a Hybrid Genetic/Random Neural Network Learning Algorithm, Pattern Analysis and Applications, 1998, 1: 52-61.
6. Aguilar J. Learning Algorithms for the Multiple Classes Random Neural Network, Lecture Notes in Artificial Intelligence, 2000, 1821: 561-566.
7. Atalay V., Gelenbe E., Yalabik N. The random neural network model for texture generation, Intl. Journal of Pattern Recognition and Artificial Intelligence, 1992, 6: 131-141.
8. Colombo C., A. Del Bimbo (1999), Color-induced image representation and retrieval, Pattern Recognition, 1999, 32: 1685-1695.
9. Del Bimbo A., Pala P. Visual image retrieval by elastic matching of user sketches, IEEE Trans. Pattern Anal. Machine Intelligence, 1997, 19: 223-234.
10. Fourneau M., Gelenbe E., Suros R. G-networks with Multiple classes of negative and positive customers, Theoretical Computer Science, 1996, 155: 141-156.
11. Gelenbe E. Random neural networks with positive and negative signals and product form solution, Neural Computation, 1989, 1: 502-511.
12. Gelenbe E. Stability of the random neural networks, Neural Computation, 1990, 2: 239-247.
13. Gelenbe E. Theory of the random neural network model, In E. Gelenbe ed., Neural Networks: Advances and Applications, North-Holland, Pays-Bas, 1991.
14. Gelenbe E. Learning in the recurrent random neural network, Neural Computation, 1993, 5: 376-389.
15. Jain A., Vailaya A. Image Retrieval using Color and Shape, Pattern Recognition, 1996, 29: 1233-1244.
16. Minka T., Picard R. Interactive Learning with a Society of Models, Pattern Recognition, 1997, 30: 1370-1381.
17. Mojsilovic A., Kovacevic J., Kall D., Safranek R., Ganapathy K. The Vocabulary and Grammar of Color Patterns, IEEE Trans. on Image Processing, 2000, 9: 417-431.

Recibido el 09 de Diciembre 2003
En forma revisada el 28 de Febrero de 2005