Statistical Pattern Recognition Problems and the Multiple Classes Random Neural Network Model

Jose Aguilar

CEMISID, Dpto. de Computación, Facultad de Ingeniería, Av. Tulio Febres,
Universidad de los Andes, Mérida 5101, 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 statistical patterns. We propose a pattern recognition algorithm for the recognition of statistical patterns based upon the non-linear equations of the multiple classes random neural network model using gradient descent of a quadratic error function. In this case the classification errors are considered.

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 pattern recognition algorithm for the recognition of statistical patterns, using MCRNN. In statistical pattern recognition a pattern is represented by a set of attributes, viewed like a *d*-dimensional feature vector [11], [12]. We present a back-propagation 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. This work is organized as follows, in section 2 the theoretical bases of MCRN are reviewed. Section 3 presents our pattern recognition algorithm for MCRNN. In section 4, we present the statistical pattern recognition problem and some comparisons. 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 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

A. Sanfeliu et al. (Eds.): CIARP 2004, LNCS 3287, pp. 336–341, 2004.

"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:

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

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 Pattern Recognition Algorithm

We propose a gradient descent learning 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:

$$\begin{aligned} w_k^+(u,p;v,z) &= w_{k-1}^+(u,p;v,c) - \mu \Sigma_{i=1}^{n} \Sigma_{c=1}^{C} (q_k(i,c) - y_k(i,c))[\delta q(i,c) / \delta w^+(u,p;v,z)]_k \\ w_k^-(u,p;v) &= w_{k-1}^-(u,p;v) - \mu \Sigma_{i=1}^{n} \Sigma_{c=1}^{C} (q_k(i,c) - y_k(i,c))[\delta q(i,c) / \delta w^-(u,p;v)]_k \end{aligned} \quad (2)$$

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 $(v\neq i)$ then

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

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

if $(u\neq i)$ 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 $(u\neq i)$ and $(v\neq i)$ 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.

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 with values equal to 0, 0.5 or 1, for each X'(i, c), i=1, ..., n and c=1, ..., C. 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 that 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:

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

where *Fz* is the thresholding function by intervals. 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)∈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 to the network the vector $X'^{(1)}(\beta)= f_\beta(Q)$. 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 much 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):

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

4 Statistical Pattern Recognition Problem

We test our approach in statistical pattern recognition problems [11]. In statistical pattern recognition a pattern is represented by a set of attributes, viewed like a *d*-dimensional feature vector [11], [12]. Well-known concepts from statistical decision theory are used to establish decision boundaries between pattern classes. The decision

making process in statistical pattern recognition can be summarized as follow: A given pattern is to be assigned to one of *c* categories w_1, ..., w_c based on a vector of *d* attributes values $x=\{x_1, ..., x_d\}$. The attributes are assumed to have a probability density or mass function condition on the pattern class (depending on whether the attributes are continuous or discrete). A number of well-known decision rules are available to define the decision boundary (Bayes decision rule, maximum likelihood rule, etc.) [11]. Various strategies are utilized to design a classifier in statistical pattern recognition according to: the kind of information available about class-conditional densities, supervised learning versus unsupervised learning, whether the decision boundaries are obtained directly (geometric approach) or indirectly (probabilistic approach). We compare our approach with several classifier systems for the recognition of a digit dataset that consists of handwritten numeral ("0"..."9") extracted from a collection of Dutch utility maps [10], [12]. Two hundred patterns per class (for a total of 2000 patterns) are available in the form of 30*48 binary images. These characters are represented in terms of the following six attributes: 76 Fourier coefficients of the character shapes, 216 profiles correlations, 64 Karhunen-Loeve coefficients, 240 pixel averages, in 2*3 windows, 47 Zernike moments, 6 morphological attributes. The following classifiers are used to compare with our approach [11]: the bayes-plug-in rule assuming normal distributions (BP), the Nearest Mean rule (NM), the Parzen technique, a feed-forward neural networks (based on the Matlab Neural Network Toolbox), with a hidden layer consisting of 20 neurons (NN), and the quadratic Support Vector classifier (SV). The classifiers were trained on the same patterns from each of the six attribute sets and tested on the same patterns. The resulting classification errors (in percentage) are reported in table 1.

Table 1. Error rate (in percentage) of different classifiers.

	Attribute Sets					
Classifiers	1	2	3	4	5	6
BP	21.3	3.4	5.7	9.9	18.9	29.1
NM	19.8	3.7	4.6	7.3	18.7	26.6
Parzen	17.1	7.9	3.7	3.7	18.5	52.1
NN	18.6	3.7	4.6	7.3	18.1	26.6
SV	21.2	5.4	4.8	6	19.3	81.1
Mult	18.2	3.5	4.2	6.2	18.6	28.2

Some of the classifiers, for example, the SV, do not perform well on this data. Some of the classifiers such as the Parzen and BP give good results. The performances of different classifiers vary substantially over different attribute sets. Only with our approach, we obtain good performances (it is not the best one) for all the attribute sets.

5 Conclusions

In this paper, we have proposed a learning algorithm based on the Multiple Classes Random Neural Model for the statistical pattern recognition problem. We have shown that this model can efficiently work as associative memory. We can learn arbitrary

statistical patterns with this algorithm. During the learning phase, we have met classical problems like the existence of local minimal 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, Vol. 4 (1996) 17-27.
3. Aguilar, J. Definition of an Energy Function for the Random Neural to solve Optimization Problems. Neural Networks, Vol. 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, Vol. 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, Vol. 6 (1992) 131-141.
6. Gelenbe, E. Random neural networks with positive and negative signals and product form solution. Neural Computation, Vol. 1 (1989) 502-511.
7. Gelenbe, E.: Stability of the random neural networks. Neural Computation, Vol. 2 (1990) 239-247.
8. Gelenbe, E.: Learning in the recurrent random neural network. Neural Computation, Vol. 5 (1993) 325-333.
9. Fourneau, M., Gelenbe, E., Suros, R.: G-networks with Multiple classes of negative and positive customers. Theoretical Computer Science, Vol. 155 (1996) 141-156.
10. Machine Learning Repository, University of California, Irvine (www.ics.ucl.edu./~mlearn/MLRepository.html)
11. Jain A., Duin R., Man J.: Statistical Pattern Recognition: A Review. IEEE Transactions on Pattern Analysis and Machine Learning, Vol. 22 (2000), 4-37.
12. Van Breukelen M., Duin R, Tax D., Den Harlog J.: Handwritten Digit Recognition by Combined Classifiers, Kybernetika, Vol. 34 (1998), 381-386.