ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 37-42

An Adaptive Technique for Pattern Recognition by the Random Neural Network

Myriam Mokhtari (*) (**)& Herman Akdag (*) (***)

(*) Leri, IUT DE REIMS, Rue des Crayères BP 257 F-51059 REIMS Cédex
(**) UFR Maths-Info, Université Paris V, 45 Rue des Saints-Pères F-75270 Paris Cédex 06
(***) Laforia, CNRS, Université Paris VI, 4 place Jussieu, BP 169, F-75252 Paris Cédex 05

e-mail : myriam@math-info.univ-paris5.fr
e-mail : akdag@laforia.ibp.fr

Abstract In this paper, we describe a random neural network model (RNNM) with positive and negative neurons which can efficiently behave like an auto-associative memory with two layers. We use the Hebb rule to compute the connection weights. To exploit the RNNM, it is necessary to know the values of the positive flow and negative flow rates entering each neuron from the outside of the network. We present here a new learning algorithm for choosing these parameters which ensure good performances for pattern recognition and pattern reconstruction.

1. Introduction

We take again the random neural network model studied in [2,3]. A neuron can emit positive signals and negative signals to its neighbours or to the outside of the network. It also can be positive or negative. There is accumulation of positive signals (respectively negative signals) at positive neurons (respectively negative neurons). Let $\underline{k}(t)$ be the vector of neuron potentials at time t and $\underline{k}=(k_1,...,k_n)$ a particular value of $\underline{k}(t)$. Following [1], if all the steady state excitation probabilities q_i are such that $0<q_i<1$, the stationary probability distribution of the network's state is expressed by : $p(\underline{k})= \Pi^n{}_{i=1}(1-q_i)q_i^{k_i}$. Let us define :

P (respectively **N**) : the set of positive (respectively negative) neurons
$\Lambda(i)$ (respectively $\lambda(i)$): the external arrival rate of positive (respectively negative) signals to i
$p^+(j,i)$ (respectively $p^-(j,i)$) : the probability that the positive neuron j emits a positive (respectively negative) signal to neuron i
$p^+(j,i)$ (respectively $p^-(j,i)$) : the probability that the negative neuron j emits a negative (respectively positive) signal to neuron i
r(i): the emission rate of signals by neuron i.

For a two layers network, the X pattern application to the network characterizes the input neurons as positive or negative :

if $X_i = 1$ then $i \in$ **P**, $\Lambda(i) \neq 0$ and $\lambda(i) = 0$

if $X_i = -1$ then $i \in$ **N**, $\lambda(i) \neq 0$ and $\Lambda(i) = 0$

the corresponding output neurons have respectively the same sign.
The q_i are computed from the following equation system :

Input layer : $q_i^0 = \frac{\Lambda(i)}{r(i)}$ if $i \in \mathbf{P}$ and $q_i^0 = \frac{\lambda(i)}{r(i)}$ if $i \in \mathbf{N}$

$$\text{Output Layer: } q_i = \frac{\Lambda(i) + \sum_{j \in \mathbf{P}} q_j^0 r(j) p^+(j,i) + \sum_{j \in \mathbf{N}} q_j^0 r(j) p^-(j,i)}{r(i) + \sum_{j \in \mathbf{P}} q_j^0 r(j) p^-(j,i) + \sum_{j \in \mathbf{N}} q_j^0 r(j) p^+(j,i)} \quad \text{if } i \in \mathbf{P}$$

$$\text{and } q_i = \frac{\lambda(i) + \sum_{j \in \mathbf{P}} q_j^0 r(j) p^-(j,i) + \sum_{j \in \mathbf{N}} q_j^0 r(j) p^+(j,i)}{r(i) + \sum_{j \in \mathbf{P}} q_j^0 r(j) p^+(j,i) + \sum_{j \in \mathbf{N}} q_j^0 r(j) p^-(j,i)} \quad \text{if } i \in \mathbf{N}$$

Thus, for each output neuron i, we will have (1):

$$q_i = \frac{\Lambda(i) + \sum_{j \in \mathbf{P}} \Lambda(j) p^+(j,i) + \sum_{j \in \mathbf{N}} \lambda(j) p^-(j,i)}{r(i) + \sum_{j \in \mathbf{P}} \Lambda(j) p^-(j,i) + \sum_{j \in \mathbf{N}} \lambda(j) p^+(j,i)} = \frac{N_i}{D_i} \quad \text{if } i \in \mathbf{P}$$

$$\text{and } q_i = \frac{\lambda(i) + \sum_{j \in \mathbf{P}} \Lambda(j) p^-(j,i) + \sum_{j \in \mathbf{N}} \lambda(j) p^+(j,i)}{r(i) + \sum_{j \in \mathbf{P}} \Lambda(j) p^+(j,i) + \sum_{j \in \mathbf{N}} \lambda(j) p^-(j,i)} = \frac{N_i}{D_i} \quad \text{if } i \in \mathbf{N}$$

with $p^+(i,j) = w_{ij} / r(i)$ if $w_{ij}>0$, $p^-(i,j) = |w_{ij}| / r(i)$ if $w_{ij}<0$
and $r(i) = \Sigma_j |w_{ij}|$, where w_{ij} are computed by the Hebb rule.

2. The Gradient Method Principle

As in general learning from noisy inputs gives better results for recognition, (noise addition avoid finding a particular solution for the vectors $\underline{\Lambda}$ and $\underline{\lambda}$, which depends on the learning set), it seems interesting to apply it to our model. Thus, the problem is to choose $\underline{\Lambda}$ and $\underline{\lambda}$ such that we obtain the desired output, meaning that for each output neuron i : $q_i \sim 0$ if the component X_i is corrupted and $q_i \sim 1$ if X_i is correct. We use the gradient descent method to minimize the quadratic error cost function E : $E = (1/2) \Sigma^n{}_{i=1}(q_i - y_i)^2$ where $y_i \in [0, 1]$ is the desired output at neurone i. The rule for $\underline{\Lambda}$ and $\underline{\lambda}$ update may be written as (2) :

$\Lambda(u) = \Lambda(u) - \eta\, \partial E/\partial\Lambda(u)$ and $\lambda(u) = \lambda(u) - \eta\, \partial E/\partial\lambda(u)$
with $\partial E/\partial\Lambda(u) = \Sigma^n{}_{i=1} (q_i - y_i) [\partial q_i/\partial\Lambda(u)]$
and $\partial E/\partial\lambda(u) = \Sigma^n{}_{i=1} (q_i - y_i) [\partial q_i/\partial\lambda(u)]$ where η is some constant.

In order to compute $\partial q_i/\partial\Lambda(u)$ and $\partial q_i/\partial\lambda(u)$ we turn to the expression (1) from which we derive the following equations (3) :
if $i \in \mathbf{N}$

$$\frac{\partial q_i}{\partial\lambda(u)} = \frac{\frac{\partial}{\partial\lambda(u)}\left[\lambda(i)+\sum_{j\in\mathbf{P}}\Lambda(j)p^-(j,i)+\sum_{j\in\mathbf{N}}\lambda(j)p^+(j,i)\right]D_i - N_i\frac{\partial}{\partial\lambda(u)}\left[\sum_{j\in\mathbf{P}}\Lambda(j)p^+(j,i)+\sum_{j\in\mathbf{N}}\lambda(j)p^-(j,i)\right]}{D_i^2}$$

$$= \frac{\frac{\partial\lambda(i)}{\partial\lambda(u)}+\sum_{j\in\mathbf{P}}\frac{\partial\Lambda(j)}{\partial\lambda(u)}p^-(j,i)+\sum_{j\in\mathbf{N}}\frac{\partial\lambda(j)}{\partial\lambda(u)}p^+(j,i)-q_i\left(\sum_{j\in\mathbf{P}}\frac{\partial\Lambda(u)}{\partial\lambda(u)}p^+(j,i)+\sum_{j\in\mathbf{N}}\frac{\partial\lambda(j)}{\partial\lambda(u)}p^-(j,i)\right)}{D_i}$$

$$= \frac{\mathbf{1}[u=i]+\mathbf{1}[u\neq i]\,\mathbf{1}[u\in\mathbf{N}]\,p^+(u,i)-q_i\,\mathbf{1}[u\neq i]\,\mathbf{1}[u\in\mathbf{N}]\,p^-(u,i)}{D_i}$$

with a similar calculus, we obtain :

$$\frac{\partial q_i}{\partial\Lambda(u)} = \frac{\mathbf{1}[u\neq i]\,\mathbf{1}[u\in\mathbf{P}]\,p^-(u,i)-q_i\,\mathbf{1}[u\neq i]\,\mathbf{1}[u\in\mathbf{P}]\,p^+(u,i)}{D_i}.$$

If $i \in \mathbf{P}$

$$\frac{\partial q_i}{\partial\Lambda(u)} = \frac{\mathbf{1}[u=i]+\mathbf{1}[u\neq i]\,\mathbf{1}[u\in\mathbf{P}]\,p^+(u,i)-q_i\,\mathbf{1}[u\neq i]\,\mathbf{1}[u\in\mathbf{P}]\,p^-(u,i)}{D_i}$$

$$\frac{\partial q_i}{\partial\lambda(u)} = \frac{\mathbf{1}[u\neq i]\,\mathbf{1}[u\in\mathbf{N}]\,p^-(u,i)-q_i\,\mathbf{1}[u\neq i]\,\mathbf{1}[u\in\mathbf{N}]\,p^+(u,i)}{D_i}$$

The learning algorithm for choosing the set of $\Lambda(i)$ and $\lambda(i)$ is the following :

1- Initialize each components of $\underline{\Lambda}$ (respectively $\underline{\lambda}$) at random values between Λ_{min} and Λ_{max} (respectively λ_{min} and λ_{max}) where Λ_{min} and Λ_{max} (respectively λ_{min} and λ_{max}) are computed by a heuristic given in [7]

2- Give as new input the vector $X = (X_1, ..., X_N)$ and its desired output $y = (y_1, ..y_N)$ (where N is the input neuron number and the output neuron number)

3- Code each X components as :
if $X_i = 1$ then $i \in \mathbf{P}$, $\Lambda(i) \neq 0$ and $\lambda(i) = 0$ else $i \in \mathbf{N}$, $\lambda(i) \neq 0$ and $\Lambda(i) = 0$

4- Compute the solution $\underline{q}$ from (1)

5- For each output neuron i compute $\partial E/\partial\Lambda(i)$ and $\partial E/\partial\lambda(i)$ with (3)

6- Update the arrival rates $\underline{\Lambda}$ and $\underline{\lambda}$ with (2):

7- Go to 2 while not convergence.

Different recall methods have been proposed in [3, 4, 6, 7]. We present here the best one. The bipolar output Y is produced as following : for each output neuron i, if $q_i \geq \alpha$ then $Y_i=X_i$ else $Y_i=-X_i$. Let us note α_0 as the minimum of the output q_i for all the stored patterns. The most efficient threshold α for the X pattern recognition is the one which generates a bipolar output Y from $\underline{q}^{(X)}$ such that when reinjecting Y to the network we have $q_i^{(Y)} \geq \alpha_0$ for each output neuron i. This method asserts the recognition of all the stored patterns. If we can not find such a threshold α, so X is still noisy. The corrupted components correspond to low q_i values (inhibited neurons). Then, the components whose q_i value is minimal are certainly corrupted. So, we start to correct these errors by this threshold function: if q_i=minq then $Y_i=-X_i$ else $Y_i=X_i$, where minq is the minimal q_i value. We reinject this output Y to the system which then becomes a new input X. After computing q_i, if $q_i \geq \alpha_0$ for $1 \leq i \leq n$, then X is corrected. If not, we correct again certain components of X as before, etc.. We stop X injection when the reinjection number is larger than n by the maximal supposed noise rate, because at worst there is only 1 correction per pass. Then, X will be the closest version of the stored pattern.

3. Experimental Results

We have stored from 1 to 16 random examples of 100 components each one. A component is equal to +1 or -1 with a probability 0.5. For each of them, 100 noisy examples have been generated with a *d %* noise rate. We have repeated the experience 10 times and the recognition performances are results of an average. Figure 1 gives the results.

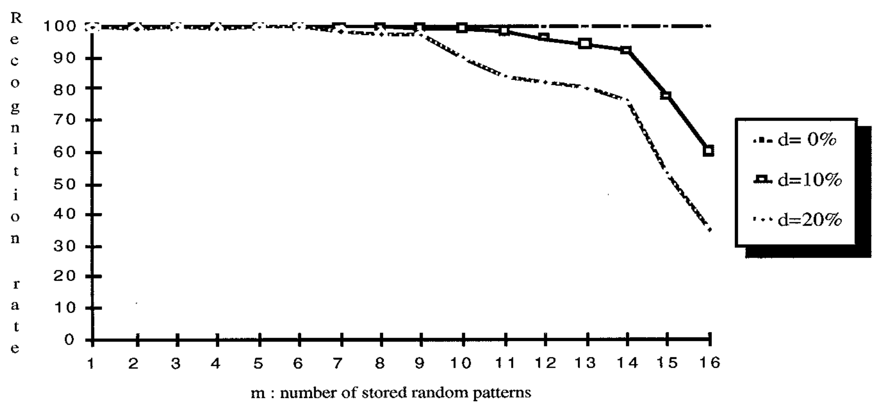

Fig. 1. Recognition rate of random patterns, where m is the number of stored patterns and d the noise rate

On the other hand, we have computed the minimal reinjection number of the pattern necessary for its exact reconstruction. Figure 2 gives us the number of the iterations with respect to the number *m* of stored patterns and the noise rate *d* .

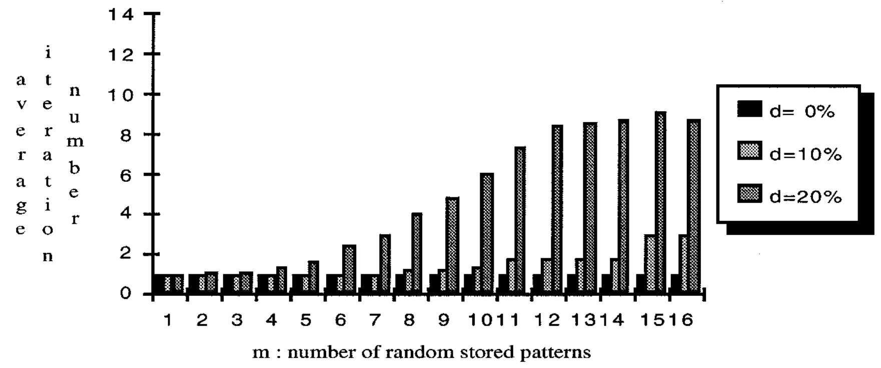

Fig. 2. Average number of iterations, where m is the number of stored patterns and d is the noise rate

4. Capacity

We define the capacity of an auto-associative memory as the maximum number of patterns that can be stored and subsequently retrieved at a given level of fidelity. For our model, we see that the stored patterns are always recalled without error, whateven their number m. That is because we have requiered a guarantee that retrieval of stored patterns be successful by choosing a threshold less than or equal to α_0. However, it is obvious that for a high number m, the patterns will not be reconstructed even with small noise. So, for the RNNM, we are interested by the maximal number of stored patterns such that no one deformation can be corrected.

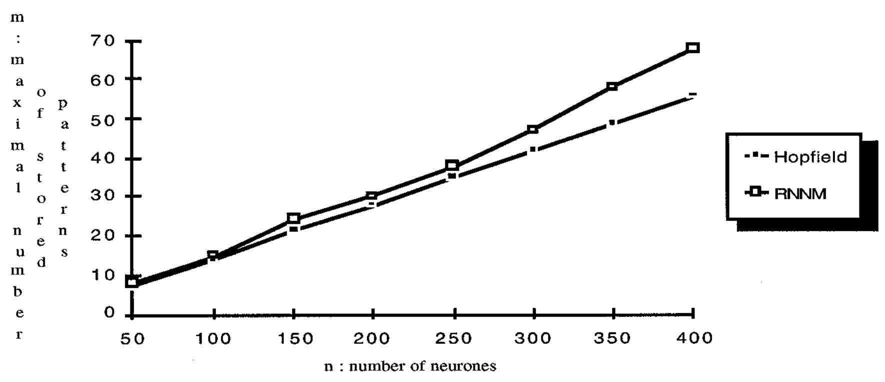

Fig. 3. Comparison between the RNNM Capacity and the Hopfield model Capacity, with respect to the neuron number n

Another manner to evaluate the capacity of the RNNM is to determine the maximal number of stored patterns such that the noisy examples with a small noise rate (for example only one corrupted component) are all reconstructed. Figure 3 gives this RNNM capacity and the one of the Hopfield model ($m \simeq 0.14\ n$).

5. Conclusion

Various modifications and refinements of our learning algorithm are worth investigating. In experiments which are not reported in the paper, we show that the RNNM provides substantially better results than Hopfield model if the stored patterns don't satisfy the Hebb rule. Finally, we can obtain better minimization and so better recognition performances if we consider that a neuron can receive at the same time positive signals and negative signals from the outside of the network.

References

1. E. Gelenbe: Random Neural Networks with negative and positive signals and product form solution. Neural Computation, Vol. 1, No. 4, 502-510 (1989).
2. E. Gelenbe, A. Stafylopatis, A. Likas: Associative memory operation of the Random Network Model. Proc. of Int. Conf. on Artificial Neural Networks, ICANN 91), Helsinki, 307-315 (1991).
3. M. Mokhtari: Storage and recognition methods for the Random Neural Network. Neural Networks:Advances and Applications II, E. Gelenbe (Editor), North-Holland, 155-176 (1992).
4. M. Mokhtari: Pattern Recognition with the Random Neural Network. Proc. of Int. Conf. on Artificial Neural Networks (ICANN92), Brighton, 1211-1214 (1992).
5. E. Gelenbe: Learning in the Recurrent Random Neural Network, Neural Networks. Advances and Applications II, E. Gelenbe (Editor), North-Holland, 1-12 (1992).
6. M. Mokhtari: "Digit recognition by the random neural network using supervised learning", Proc. International Conference on Artificial Neural Networks (ICANN 93), Amsterdam, Pays-Bas, Septembre 1993.
7. M. Mokhtari:"Réseau Neuronal Aléatoire : Applications à l'Apprentissage et à la Reconnaissance d'images". Thèse de Doctorat, Université René Descartes, Paris V, Janvier 1994.
8. M. Mokhtari: "Classification by the Random Neural Network", Proc. Tenth International Conference on Mathematical and Computer Modelling and Scientific Computing (10th ICMCM), Boston, U.S.A., Juillet 1995.