ESANN'1993 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 7-8-9 April 1993, D-Facto public., ISBN 2-9600049-0-6, pp. 171-176

SUPERVISED LEARNING AND ASSOCIATIVE MEMORY BY THE RANDOM NEURAL NETWORK

M. Mokhtari

Ecole des Hautes Etudes en Informatique, Université René Descartes, 45 rue des Saints Pères, 75006 Paris, France

Abstract The Random Neural Network model with positive and negative neurons is studied as an autoassociative memory for pattern recognition. We first apply the learning algorithm for the recurrent random network model (Gelenbe 91) to the recurrent random network with positive and negative neurons. Then, we implement this learning algorithm for a particular case of network. We also present a fast recognition algorithm based on model properties and which give a very good performance for digit recognition.

1. Introduction

In [1,2], a random neural network model has been introduced. A neuron can excite its neighbours if it emits positive signals or can inhibit them by emission of negative signals, and all neurons are of the same type. The model has been extended in [3] by giving a sign to each neuron. Here we consider this model. According to [3], there is accumulation of positive signals at positive neurons and negative signals at negative neurons. Neuron potential is the number of accumulated signals. The arrival of a positive signal to a positive neuron adds 1 to its potential, whereas it reduces by 1 that of negative neuron or has no effect if the neuron potential is already zero. Symmetrically, the arrival of a negative signal to a negative neuron adds 1 to its potential, whereas it reduces by 1 that of positive neuron or has no effect if the neuron potential is already zero. When the potential of a neuron i is strictly positive, the neuron is said to be excited, so it can emit signals at random intervals with exponential interfiring times of constant rate r(i). A signal which leaves a positive neuron i heads for neuron j with probability $p^+(i,j)$ as a positive signal, or $p^-(i,j)$ as a negative one. Symmetrically, a signal which leaves a negative neuron i heads for neuron j with probability $p^+(i,j)$ as a negative signal, or $p^-(i,j)$ as a positive one. External positive (respectively negative) signals arrive to the i-th positive (respectively negative) neuron according to a Poisson process of rate $\Lambda(i)$ (respectively $\lambda(i)$). Let $\underline{k}(t)$ be the vector of neuron potentials at time t, and $\underline{k}=(k_1,...,k_n)$ be a particular value. It is proved in [2] that, 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 given by $p(\underline{k})=lim_{t->\infty}P[\underline{k}(t)=\underline{k}]$ exists and can be expressed by: $p(\underline{k})=\prod^n_{i=1}(1-q_i)q_i^{k_i}$. The q_i are computed from the following system of non-linear equations (1-3) where **P** (respectively **N**) is the set of positive (respectively negative) neurons:

$$q_i = \gamma^+(i) / [r(i)+\gamma^-(i)] \text{ if } i\in \mathbf{P},\quad q_i = \gamma^-(i) / [r(i)+\gamma^+(i)] \quad \text{if } i\in \mathbf{N} \qquad (1)$$

with: $\gamma^+(i) = \Lambda(i) + \Sigma_{j\in \mathbf{P}}\ q_j\, r(j)p^+(j,i) + \Sigma_{j\in \mathbf{N}}\ q_j\, r(j)p^-(j,i)$ (2)

$\gamma^-(i) = \lambda(i) + \Sigma_{j\in \mathbf{P}}\ q_j\, r(j)p^-(j,i) + \Sigma_{j\in \mathbf{N}}\ q_j\, r(j)p^+(j,i)$ (3)

In recent papers [4,5,6,7], we have shown that the random neural network model with positive neurons and negative neurons can efficiently work as an auto-associative memory whatever the correlation between the stored patterns and the pattern to be recognized. The Hopfield architecture for the network and the Hebbian rule for learning have been chosen. Here, we use a supervised learning based on gradient descent of a quadratic error function. This learning method has been presented by Erol Gelenbe [8] for the recurrent (i.e. general) random network model. We first apply it to the recurrent network with positive and negative neurons, and then use it as a particular case for a network with only 2 layers. The network is then tested for the digit recognition.

2. Learning with the recurrent random network model with positive and negative neurons

In what follows, we take Gelenbe's notations in [8] in order to apply its learning algorithm to the random neural network with positive and negative neurons. This algorithm is used for choosing the network parameters r(i), $p^+(i,j)$ and $p^-(i,j)$ in order to learn a given set of K input-output pairs $(\iota,\mathbf{Y})$ where the set of successive inputs is denoted $\iota= \{\iota_1,...,\iota_K\}$ and $\iota_k=(\Lambda_k,\lambda_k)$ are pairs of positive and negative signal flow rates entering each neuron: $\Lambda_k=(\Lambda_k(1),...,\Lambda_k(n))$, $\lambda_k=(\lambda_k(1),...,\lambda_k(n))$. The successive desired outputs are the vectors $\mathbf{Y}= \{y_1,...,y_K\}$, where each vector $y_k=\{y_{1k},...,y_{nk}\}$, whose elements $y_{ik}\in [0,1]$ correspond to the desired values of each neuron. The network approximates the set **Y** such that the cost function $E_k=(1/2)\ \Sigma^n_{i=1}a_i(q_{ik}-y_{ik})$ is minimized, where a_i is the contribution of neuron i. Let us write

$w^+(i,j)=r(i)p^+(i,j) \geq 0$, $w^-(i,j)=r(i)p^-(i,j) \geq 0$, $r(i)=\Sigma_j\ [w^+(i,j)+w^-(i,j)]$, and

if $i\in \mathbf{P}$ $N(i)=\Lambda(i) + \Sigma_{j\in \mathbf{P}}\ q_j\, r(j)p^+(j,i) + \Sigma_{j\in \mathbf{N}}\ q_j\, r(j)p^-(j,i)$

$D(i)=r(i) + \lambda(i) + \Sigma_{j\in \mathbf{P}}\ q_j\, r(j)p^-(j,i) + \Sigma_{j\in \mathbf{N}}\ q_j\, r(j)p^+(j,i)$,

if $i\in \mathbf{N}$ $N(i)=\lambda(i) + \Sigma_{j\in \mathbf{P}}\ q_j\, r(j)p^-(j,i) + \Sigma_{j\in \mathbf{N}}\ q_j\, r(j)p^+(j,i)$

$D(i)=r(i) + \Lambda(i) + \Sigma_{j\in \mathbf{P}}\ q_j\, r(j)p^+(j,i) + \Sigma_{j\in \mathbf{N}}\ q_j\, r(j)p^-(j,i)$,

then (1) becomes $q_i=N(i)/D(i)$, (4)

The algorithm lets the network learn both n by n matrices $W^+{}_k= \{w^+{}_k(i,j)\}$ and $W^-{}_k= \{w^-{}_k(i,j)\}$ by computing for each input $\iota_k=(\Lambda_k,\lambda_k)$, a new value $W^+{}_k$ and $W^-{}_k$ of the weights matrices, using gradient descent. Let us denote by the generic term w(u,v) either $w(u,v)\equiv w^-(u,v)$ or $w(u,v)\equiv w^+(u,v)$. The rule for weight update may be written as : $w_k(u,v)=w_{k-1}(u,v) -\eta\ \Sigma^n_{i=1}\ a_i\ (q_{ik}-y_{ik})\ [\partial q_i/\partial w(u,v)]_k$ (5)

where η is some constant, and :

(i) q_{ik} is calculated using the input ι_k and $w(u,v)=w_{k-1}(u,v)$, in equation (4),

(ii) $[\partial q_i/\partial w(u,v)]_k$ is evaluated at the values $q_i=q_{ik}$ and $w(u,v)=w_{k-1}(u,v)$.

In order to compute $[\partial q_i/\partial w(u,v)]_k$, we turn to the expression (4), from which we derive the following equation :

if $i \in \mathbf{P}$: $\partial q_i/\partial w(u,v)=\Sigma_j\, \partial q_j/\partial w(u,v)\ [\ (\ \mathbf{1}[j\in \mathbf{P}]\ w^+(j,i) + \mathbf{1}[j\in \mathbf{N}]\ w^-(j,i)\ )$
$-(\mathbf{1}[j\in \mathbf{P}]\ w^-(j,i) + \mathbf{1}[j\in \mathbf{N}]\ w^+(j,i)\ )q_i\ ]\ /\ D(i)$
$-\mathbf{1}[u=i]\ q_i/D(i) + \mathbf{1}[w(u,v)\equiv w^+(u,i)]\ (\mathbf{1}[u\in \mathbf{P}]\ -\mathbf{1}[u\in \mathbf{N}]q_i)q_u/D(i)$
$+\mathbf{1}[w(u,v)\equiv w^-(u,i)]\ (\mathbf{1}[u\in \mathbf{N}]\ -\ \mathbf{1}[u\in \mathbf{P}]q_i)q_u/D(i)$

if $i \in \mathbf{N}$: $\partial q_i/\partial w(u,v)=\Sigma_j\, \partial q_j/\partial w(u,v)\ [\ (\ \mathbf{1}[j\in \mathbf{N}]\ w^+(j,i) + \mathbf{1}[j\in \mathbf{P}]\ w^-(j,i)\ )$
$-\ (\ \mathbf{1}[j\in \mathbf{N}]\ w^-(j,i) + \mathbf{1}[j\in \mathbf{P}]\ w^+(j,i)\ )q_i\ ]\ /\ D(i)$
$-\mathbf{1}[u=i]\ q_i/D(i) + \mathbf{1}[w(u,v)\equiv w^+(u,i)]\ (\mathbf{1}[u\in \mathbf{N}]\ -\mathbf{1}[u\in \mathbf{P}]q_i)q_u/D(i)$
$+\mathbf{1}[w(u,v)\equiv w^-(u,i)]\ (\mathbf{1}[u\in \mathbf{P}]\ -\ \mathbf{1}[u\in \mathbf{N}]q_i)q_u/D(i).$

We can write : (6) $$\begin{cases} \partial q_i/\partial w^+(u,v)= \Sigma_j\, \partial q_j/\partial w(u,v)\ W(j,i) + \gamma^+_i(u,v)q_u \\ \partial q_i/\partial w^-(u,v)= \Sigma_j\, \partial q_j/\partial w(u,v)\ W(j,i) + \gamma^-_i(u,v)q_u \end{cases}$$

where $\gamma^+_i(u,v)$, $\gamma^-_i(u,v)$ and $W(j,i)$ are for $i,j=1,...,n$:

if $i \in \mathbf{P}$: $W(j,i) = [\ w^+(j,i)(\mathbf{1}[j\in \mathbf{P}] - \mathbf{1}[j\in \mathbf{N}]q_i) + w^-(j,i)(\mathbf{1}[j\in \mathbf{N}] - \mathbf{1}[j\in \mathbf{P}]q_i)\ ]/D(i)$
$\gamma^+_i(u,v) = -1/D(i)$ if $u=i,\ v\neq i$
$= (\mathbf{1}[u\in \mathbf{P}] - \mathbf{1}[u\in \mathbf{N}]q_i)/D(i)$ if $v=i,\ u\neq i$
$= 0$ for all other values of (u,v)
$\gamma^-_i(u,v) = -1/D(i)$ if $u=i,\ v\neq i$
$= (\mathbf{1}[u\in \mathbf{N}] - \mathbf{1}[u\in \mathbf{P}]q_i)/D(i)$ if $v=i,\ u\neq i$
$= -(1+q_i)/D(i)$ if $u=i,\ v=i$
$= 0$ for all other values of (u,v)

if $i \in \mathbf{N}$: $W(j,i) = [\ w^+(j,i)(\mathbf{1}[j\in \mathbf{N}] - \mathbf{1}[j\in \mathbf{P}]q_i) + w^-(j,i)(\mathbf{1}[j\in \mathbf{P}] - \mathbf{1}[j\in \mathbf{N}]q_i)\ ]/D(i)$
$\gamma^+_i(u,v) = -1/D(i)$ if $u=i,\ v\neq i$
$= (\mathbf{1}[u\in \mathbf{N}] - \mathbf{1}[u\in \mathbf{P}]q_i)/D(i)$ if $v=i,\ u\neq i$
$= 0$ for all other values of (u,v)
$\gamma^-_i(u,v) = -1/D(i)$ if $u=i,\ v\neq i$
$= (\mathbf{1}[u\in \mathbf{P}] - \mathbf{1}[u\in \mathbf{N}]q_i)/D(i)$ if $v=i,\ u\neq i$
$= -(1+q_i)/D(i)$ if $u=i,\ v=i$
$= 0$ for all other values of (u,v).

The learning algorithm for the network is as follows:
- initialize the matrix W^+_0 and W^-_0 in some apropriate manner.
- choose a value of the learning rate η in (5).
(i) For each successive value of k, starting with k=1, proceed as follows. Set the input values to $\iota_k=(\Lambda_k,\lambda_k)$.
(ii) Solve the system of non-linear equations (4) with these values.
(iii) Solve the system of non linear equations (6) with the results of (ii).
(iv) Using (5) and the results of (ii) and (iii), update the matrices W^+_k and W^-_k.

3. Random neural network as an auto-associative memory

3.1. Network architecture and parameters

Realization of an auto-associative memory consists of learning a set of patterns and making the network recognize them, even if they are noisy. In this last case, the recognition means pattern reconstruction. Let us suppose we have K patterns of dimension n, and note the set by $X=\{x_1,...,x_K\}$, where $x_k=\{x_{1k},...,x_{nk}\}$, with $x_{ik} \in \{-1,1\}$, and x_{ik} is the k^{th} input of neuron i. So, n neurons are necessary for the input layer. We also take n neurons for the output layer. The network architecture is given by Fig. 1 where n=6 for exemple.

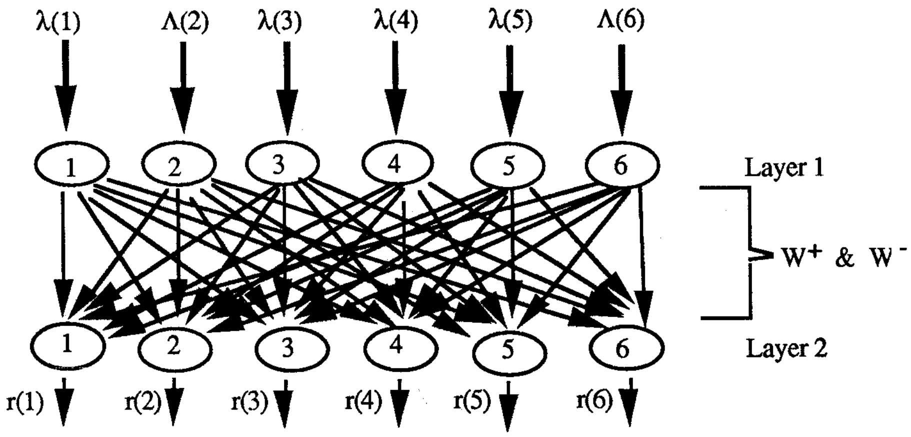

Fig. 1. Network architecture

When a pattern x_k is applied as input to the network, it characterizes network neurons as positive or negative : if x_{ik}='-1' then $i \in \mathbf{N}$, $\lambda(i)=\lambda_k(i)$ and $\Lambda(i)=0$; if x_{ik}='1' then $i \in \mathbf{P}$, $\Lambda(i)=\Lambda_k(i)$ and $\lambda(i)=0$. We take the following assumptions: $\lambda_k(i)=\Lambda_k(i)=1000$ for $i \in$ Layer 1 such that all the input neurons are saturated ($q_{ik}>1$; if $q_{ik}>1$ set $q_{ik}=1$); and $r(j)=r(i)$ for $i \in$ Layer 1 and $j \in$ Layer 2. Let us remind that $r(i)=\sum_j[w^+(i,j)+w^-(i,j)]$ for $i \in$ Layer 1 and $j \in$ Layer 2.

3.2. Learning algorithm

Following the previous assumptions, whatever the input x_k we have for the input layer Layer 1 : $q_{ik}=\lambda_k(i)/r(i)=1$ if $i \in \mathbf{N}$ and $q_{ik}=\Lambda_k(i)/r(i)=1$ if $i \in \mathbf{P}$. So we have $\partial q_i/\partial w(u,v)=0$ for $i \in$ Layer 1 (with $u \in$ Layer 1 and $v \in$ Layer 2). The system of non linear equations (6) is then simplified to

$\partial q_i/\partial w^+(u,v)=\gamma^+_i(u,v)$ and $\partial q_i/\partial w^-(u,v)=\gamma^-_i(u,v)$ for $i \in$ Layer 2 (7)

where $\gamma^+_i(u,v)$, $\gamma^-_i(u,v)$ are:

if $i \in \mathbf{P}$:

$$\begin{aligned}\gamma^+_i(u,v) &= -q_i/D(i) \text{ if } u=i, v\neq i\\ &= (1[u\in \mathbf{P}] - 1[u\in \mathbf{N}]q_i)/D(i) \text{ if } v=i, u\neq i\\ &= (1-q_i)/D(i) \text{ if } u=i, v=i\\ &= 0 \text{ for all other values of } (u,v)\end{aligned}$$

$$\begin{aligned}\gamma^-_i(u,v) &= -q_i/D(i) \text{ if } u=i, v\neq i\\ &= (1[u\in \mathbf{N}] - 1[u\in \mathbf{P}]q_i)/D(i) \text{ if } v=i, u\neq i\\ &= -2q_i/D(i) \text{ if } u=i, v=i\\ &= 0 \text{ for all other values of } (u,v)\end{aligned}$$

if $i \in \mathbf{N}$:

$$\begin{aligned}\gamma^+_i(u,v) &= -q_i/D(i) \text{ if } u=i, v\neq i\\ &= (1[u\in \mathbf{N}] - 1[u\in \mathbf{P}]q_i)/D(i) \text{ if } v=i, u\neq i\\ &= (1-q_i)/D(i) \text{ if } u=i, v=i\\ &= 0 \text{ for all other values of } (u,v)\end{aligned}$$

$$\begin{aligned}\gamma^-_i(u,v) &= -q_i/D(i) \text{ if } u=i, v\neq i\\ &= (1[u\in \mathbf{P}] - 1[u\in \mathbf{N}]q_i)/D(i) \text{ if } v=i, u\neq i\\ &= -2q_i/D(i) \text{ if } u=i, v=i\\ &= 0 \text{ for all other values of } (u,v)\end{aligned}$$

The learning algorithm is described following the four steps from (i) to (iv) of section 2. In the cost function E_k we take $a_i=1$ for $i\in$ Layer 2 since every neuron contributes to the error cost, and the desired output y_{ik} is always set to 1 since all neurons must be excited (q_{ik} close to 1) for each $x_k\in X$. Let us denote by α_0 the minimal q_i value on all neurons and all stored patterns once the learning stage is achieved: $\alpha_0 = \min_{1\leq k\leq K} (\min_{1\leq i\leq n} q_{ik})$. This value which is close to 1 defines the storage threshold. It will be useful for the recognition stage.

3.3. Recognition algorithm

When the learning stage is finished, each pattern x to be recognized is coded as: for $1\leq i\leq n$, if $x_i=1$, then $i\in \mathbf{P}$, $\Lambda(i)=1000$ and $\lambda(i)=0$; if $x_i=-1$, then $i\in \mathbf{N}$, $\lambda(i)=1000$ and $\Lambda(i)=0$. After computing the q_i, the x recognition is defined as follows: if the computed q_i are such that $q_i\geq\alpha_0$ for $1\leq i\leq n$, the pattern x is a stored one otherwise it is noisy: the corrupted components correspond to low q_i values (inhibited neurons). Then, the components whose q_i value is minimal are certainly corrupted (others can exist). 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 x_k version.

4. Results

The recognition has been applied to the 10 decimal numerals shown in Fig. 2. Each digit is coded on 64 (8 by 8) components (so n=64 for Layer 1 and for Layer 2).

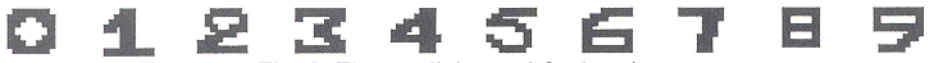

Fig. 2. The ten digits used for learning

For the learning stage, we have taken $\eta = 0.2$ in the cost function E_k and we have initialized the matrices W^+_0 and W^-_0 at random with small values between 0 and 1. Weights have been modified so as to decrease the cost function E_k until 0.0032. We have corrupted the input patterns by a noise rate which could reach 20%. A recognition test was considered successful if the output exactly corresponds to the digit we want to recall. Recognition performance is given by Fig. 3.

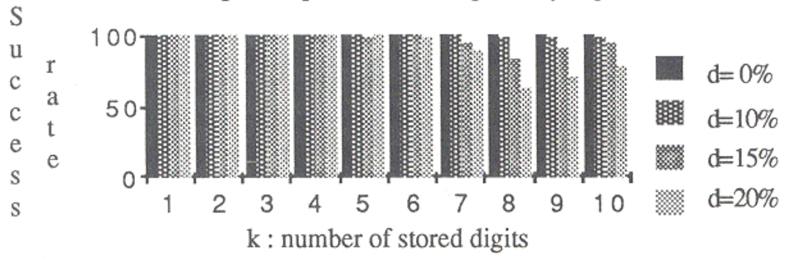

Fig. 3. Recognition performance as a function of the number of stored digits and the noise level

Almost all patterns whose noise rate d is less than or equal to 15% are recognized, whatever the number of stored patterns. Even for an important noise rate (d=20%), the recognition is good.

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: Stability of the Random Neural Network Model. Neural Computation, vol.2, No. 2, 239-247 (1990)
3. 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)
4. M. Mokhtari: Recognition of schematic images by the Random Neural Network. Proc. of International Colloq. on Parallel Image Processing, Paris, 205-211 (1991)
5. 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)
6. M. Mokhtari: Pattern Recognition with the Random Neural Network. Proc. of Int. Conf. on Artificial Neural Networks (ICANN92), Brighton, 1211-1214 (1992)
7. M. Mokhtari: The Random Neural Network with some saturated neurons for Auto-associative memory. Proc. of International Symposium on Computer and Information Sciences (ISCIS VII), Antalya, 585-588 (1992)
8. E. Gelenbe: Learning in the Recurrent Random Neural Network, Neural Networks. Advances and Applications II, E. Gelenbe (Editor), North-Holland, 1-12 (1992).