Random Neural Networks with Negative and Positive Signals and Product Form Solution

Article *in* Neural Computation · December 1989

DOI: 10.1162/neco.1989.1.4.502

Erol Gelenbe
Polish Academy of Sciences
772 PUBLICATIONS **18,377** CITATIONS

Communicated by Jack Cowan

Random Neural Networks with Negative and Positive Signals and Product Form Solution

Erol Gelenbe
Ecole des Hautes Etudes en Informatique (EHEI), Université Paris V, 45 rue des Saints-Pères, 75006 Paris, France

We introduce a new class of random "neural" networks in which signals are either negative or positive. A positive signal arriving at a neuron increases its total signal count or potential by one; a negative signal reduces it by one if the potential is positive, and has no effect if it is zero. When its potential is positive, a neuron "fires," sending positive or negative signals at random intervals to neurons or to the outside. Positive signals represent excitatory signals and negative signals represent inhibition. We show that this model, with exponential signal emission intervals, Poisson external signal arrivals, and Markovian signal movements between neurons, has a product form leading to simple analytical expressions for the system state.

1 Introduction

Consider an open random network of n neurons in which "positive" and "negative" signals circulate. External arrivals of signals to the network can either be positive, arriving at the ith neuron according to a Poisson process of rate $\Lambda(i)$, or negative according to a Poisson process of rate $\lambda(i)$. Positive and negative signals have opposite roles. A negative signal *reduces by 1* the potential of the neuron to which it arrives (i.e., it "cancels" an existing signal) or has no effect if the potential is zero. A positive signal *adds 1* to the neuron potential. Negative potentials are not allowed at neurons. If the potential at a neuron is positive, it may "fire," sending signals out toward other neurons or to the outside of the network. As signals are sent, they deplete the neuron's potential by the same number. The times between successive signal emissions when neuron i fires are exponentially distributed random variables of average value $1/r(i)$; hence $r(i)$ is the rate at which neuron i fires.

A signal leaving neuron i when it "fires" heads for neuron j with probability $p^+(i,j)$ as a positive signal, or as a negative signal with probability $p^-(i,j)$, or departs from the network with probability $d(i)$. Let $p(i,j) = p^+(i,j) + p^-(i,j)$; it is the transition probability of a Markov chain representing the movement of signals between neurons. We shall not allow the signals leaving a neuron to return directly back to the same neuron: $p(i,i) = 0$ for all i. We have

$$\sum_j p(i,j) + d(i) = 1 \quad \text{for } 1 \leq i \leq n \tag{1.1}$$

Positive signals represent excitation and negative signals represent inhibition. Positive external arrivals represent input information and negative external arrivals can be used to represent the thresholds at each neuron. A simple example of the computational use of this model is presented in Section 3.

We show that this new model has "product form" solution (Gelenbe and Mitrani 1980; Gelenbe and Pujolle 1986). That is, the stationary probability distribution of its state can be written as the product of the marginal probabilities of the state (or potential) of each neuron. This leads to simple expressions for the network state. Previously, product form solutions were known to exist for certain networks with only "positive signals," which are queueing networks used in computer and communication system modeling and in operations research (Gelenbe and Mitrani 1980; Gelenbe and Pujolle 1986).

2 The Main Properties of the Model

The main properties of our model are presented in the following theorems.

Theorem 1. *Let q_i denote the quantity*

$$q_i = \lambda^+(i)/[r(i) + \lambda^-(i)] \tag{2.1}$$

where the $\lambda^+(i), \lambda^-(i)$ for $i = 1, \ldots, n$ satisfy the system of nonlinear simultaneous equations:

$$\lambda^+(i) = \sum_j qjr(j)p^+(j,i) + \Lambda(i), \lambda^-(i) = \sum_j qjr(j)p^-(j,i) + \lambda(i) \tag{2.2}$$

Let $k(t)$ be the vector of neuron potentials at time t, and $k = (k_1, \ldots, k_n)$ be a particular value of the vector; let $p(k)$ denote the stationary probability distribution

$$p(k) = \lim_{t\to\infty} Prob[k(t) = k]$$

If a nonnegative solution $\{\lambda^+(i), \lambda^-(i)\}$ exists to equations 2.1 and 2.2 such that each $q_i < 1$, then

$$p(k) = \prod_{i=1}^{n} [1 - q_i] q_i^{k_i} \tag{2.3}$$

The proof is given in Appendix A. A direct consequence is:

Corollary 1.1 The stationary probability that a neuron i fires is given by

$$\lim_{t\to\infty} \text{Prob}[k_i(t) > 0] = q_i = \lambda^+(i)/[r(i) + \lambda^-(i)] \quad \text{if } q_i < 1$$

2.1 Networks with Some Saturated Neurons. We say that neuron i is *saturated* if $\lambda^+(i)/[r(i)+\lambda^-(i)] \geq 1$; i.e., in steady-state it continuously fires. In many applications, one is interested in working with networks containing some saturated neurons.

We have the following extension of Theorem 1. Let NS be the (largest) subset of neurons such that no neuron in NS is saturated, and S be its complement. Consider the solutions $\lambda^+(i), \lambda^-(i)$ of the flow equations:

$$\lambda^+(i) = \sum_j qjr(j)p^+(j,i) + \Lambda(i), \quad \lambda^-(i) = \sum_j qjr(j)p^-(j,i) + \lambda(i),$$

where

$$q_i = \lambda^+(i)/[r(i)+\lambda^-(i)], \quad \text{if } i \in NS \text{ and } q_i = 1 \text{ if } i \in S$$

Theorem 2. *Let $k(t)_{NS}$ denote the restriction of the vector $k(t)$ to the neurons in NS. If a positive solution to the flow equations exists then*

$$\begin{aligned} \lim\nolimits_{t\to\infty} P[k_i(t) > 0] &= \lambda^+(i)/[r(i)+\lambda^-(i)], && \text{if } i \in NS \\ & 1, && \text{if } i \in S \end{aligned}$$

and

$$\lim_{t\to\infty} P[k_{NS}(t) = k_{NS}] = p(k_{NS}) = \prod_{i \in NS} [1-q_i]q_i^{k_i}$$

We omit the proof of this result.

2.2 Equations 2.1 and 2.2 Describing Signal Flow in Feedforward Networks. Let us now turn to the existence and uniqueness of the solutions $\lambda^+(i)$, $\lambda^-(i)$, $1 \leq i \leq n$ to equations 2.1 and 2.2, which represent the average arrival rate of positive and negative signals to each neuron. We are unable to guarantee the existence and uniqueness of these quantities for arbitrary networks, except for *feedforward networks.*

A network is said to be *feedforward* if for any sequence $i_1, \ldots, i_s, \ldots, i_r, \ldots, i_m$ of neurons, $i_s = i_r$ for $r > s$ implies that

$$\prod_{v=1}^{m-1} p(i_v, i_{v+1}) = 0$$

Theorem 3. *If the network is feedforward, then the solutions $\lambda^+(i)$, $\lambda^-(i)$ to equations 2.1 and 2.2 exist and are unique.*

Proof. For any feedforward network, we may construct an isomorphic network by renumbering the neurons so that neuron 1 has no predecessors [i.e., $p(i,1) = 0$ for any i], neuron n has no successors [i.e., $p(n,i) = 0$ for any i], and $p(i,j) = 0$ if $j < i$.

Thus in the isomorphic network, a signal can possibly (but not necessarily) go directly from neuron i to neuron j only if j is larger than i.

For such a network, the $\lambda^+(i)$ and $\lambda^-(i)$ can be calculated recursively as follows:

First compute $\lambda^+(1) = \Lambda(1), \lambda^-(1) = \lambda(2)$, and calculate q_1 from equation 2.1; if you obtain $q_1 \geq 1$, set $q_1 = 1$ (neuron 1 is saturated), otherwise leave it unchanged.

For each successive i such that $\lambda^+(i), \lambda^-(i)$ have not yet been calculated proceed as follows; since the q_j for each $j < i$ are known, compute

$$\lambda^+(i) = \sum_{j<i} qjr(j)p^+(j,i) + \Lambda(i), \quad \lambda^-(i) = \sum_{j<i} qjr(j)p^-(j,i) + \lambda(i)$$

and then compute q_i; if $q_i \geq 1$ replace it by $q_i = 1$; otherwise leave it unchanged. The procedure will end because at each step the computations are carried out for an increased value of i.

This completes the proof since we have proved existence and uniqueness by calculating in a unique manner the solution to equations 2.1 and 2.2 for a feedforward network. QED

3 Analogy with Formal Neural Networks

A formal neural network (Rumelhart *et al.* 1986) is a set of n neurons each of which computes its *state* $y(i)$ using a sigmoid function $y(i) = f(x(i))$ where $x(i) = (\sum_j w_{ji}y(j) - \theta_i)$ is the *input signal* composed of the weighted sums of the states $y(j)$ of the other neurons of the network; the w_{ji} are the weights and θ_i is the threshold. In its simplest form $f(\cdot)$ is the unit step function. The set of weights and the set of thresholds completely characterize the network.

An analogy between the usual model of neural networks and the model introduced in this paper, which we henceforth call *the random network*, can be constructed. Let us remain within the framework of feedforward networks, of which *multilayer formal neural networks* (Rumelhart *et al.* 1986) are a subclass, for which Theorem 3 has been proved.

Each neuron is represented by a neuron of the random network. The threshold of neuron i is represented by a flow of negative signals to the neuron so that $\lambda(i) = \theta_i$.

Consider the nonoutput neuron i; it is represented by the random neuron i whose parameters are chosen as follows:

$$d(i) = 0, \quad r(i)p^+(i,j) = w_{ij} \text{ if } w_{ij} > 0 \text{ and } r(i)p^-(i,j) = |w_{ij}| \text{ if } w_{ij} < 0$$

Summing over all j, the firing rate $r(i)$ of the nonoutput "random" neuron i is chosen:

$$r(i) = \sum_j |w_{ij}|$$

Finally, for each output random neuron i set $d(i) = 1$, and assign some appropriate value to $r(i)$.

To introduce into the random network the external parameters or inputs to the neural network we use the arrival rates of positive signals $\Lambda(i)$. Assume that the external signals to the formal neural network are binary. If the input signal to neuron i is 0 or if neuron i is not connected to an external signal we set $\Lambda(i) = 0$. If the external signal to neuron i has the value 1, we set $\Lambda(i) = \Lambda$. Here Λ can be chosen so as to obtain the desired effect at the output neurons.

All the parameters of the random network are chosen from those of the formal network, except for the firing rates of the output neurons, and the input rate of positive signals.

The state $Y = (y_1, \ldots, y_n)$ of the formal neural network, where y_i is 0 or 1, is simulated by the probabilities of the random neurons.

Let $z_i = \lim t \to \infty P[k_i(t) > 0]$; we associate y_i to z_i. Thus, we have for some "cut-point" $1 - \alpha$,

$$[y_i = 0] \Longleftrightarrow z_i < 1 - \alpha; \quad [y_i = 1] \Longleftrightarrow z_i \geq 1 - \alpha$$

In an arbitrary neural network, that is, one that does not have the feed-forward structure, this procedure could also be used for establishing the random network. We could also use the $d(i)$ at each neuron i to represent the loss currents that are known to exist in biological neural systems (Kandel and Schwartz 1985), and take $r(i)d(i)$ to be the rate of loss of electric potential at a neuron if we wish to include this effect in the model.

3.1 An Example: The Neural Network for the XOR Function. A simple example often given to illustrate the behavior of formal neural networks is the network for the XOR (exclusive OR) function (Rumelhart *et al.* 1986). We present the equivalent random model representation for this network.

In Figure 1 we show a formal neural network that receives two binary inputs x_1, x_2 and that produces $y(x_1, x_2)$ the boolean function XOR. It is composed of four neurons; each neuron is numbered (1 to 4) and the synaptic weights are indicated on arcs between neurons. The threshold of each neuron is assumed to be 0.

In Figure 2 we show the random network analog corresponding to Figure 1. According to the rules we have given for constructing the random network analog, we have

$\lambda(i) = 0$ for $i = 1, \ldots, 4$ because all thresholds are 0;

$r(1) = r(2) = 2$, $r(3) = 1.1$, $r(4) = r$, as yet undetermined;

$p^+(1,3) = p^+(2,3) = 0.5$, $p^-(1,4) = p^-(2,4) = 0.5$, $p^+(3,4) = 1$;

$d(1) = d(2) = d(3) = 0$, $d(4) = 1$.

Recall that according to the rules proposed in Section 3, we choose a value $\Lambda(i) = \Lambda$ to represent $x_i = 1$, and $\Lambda(i) = 0$ to represent $x_i = 0$. Set

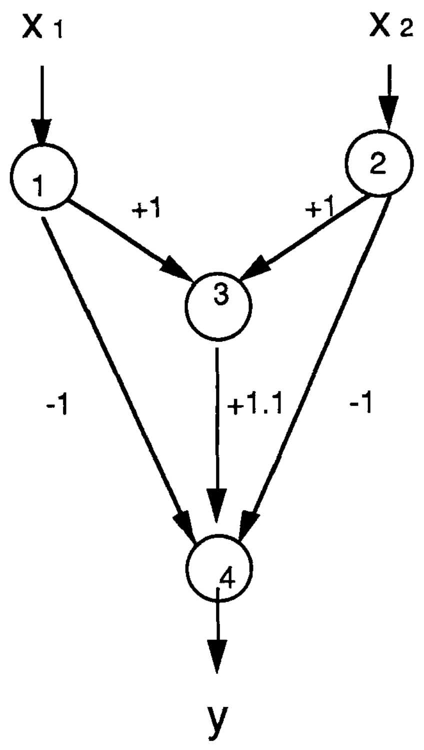

Figure 1: A formal neural network for the boolean XOR function.

Λ large enough to saturate neurons 1 and 2, that is, any $\Lambda > 2$. z_4 is the analog of the output y of the neural network of Figure 1. Notice that

$$z_4 = \begin{cases} 0, & \text{if } \Lambda(1) = \Lambda(2) = 0 \\ 1.1/[r+2], & \text{if } \Lambda(1) = \Lambda(2) = \Lambda > 2 \\ 1/[r+1], & \text{if } \Lambda(1) = \Lambda, \Lambda(2) = 0 \text{ or vice versa} \end{cases}$$

Setting $\alpha = 0.6$ and $r = 0.1$ we see that we obtain the XOR function with this network. In fact we may choose any $1 - \alpha$ such that $1.1/[r+2] < 1 - \alpha < 1/[r+1]$.

4 Conclusions

We have introduced a new random neural network in which "negative" and "positive" signals circulate among "neurons." Positive signals ac-

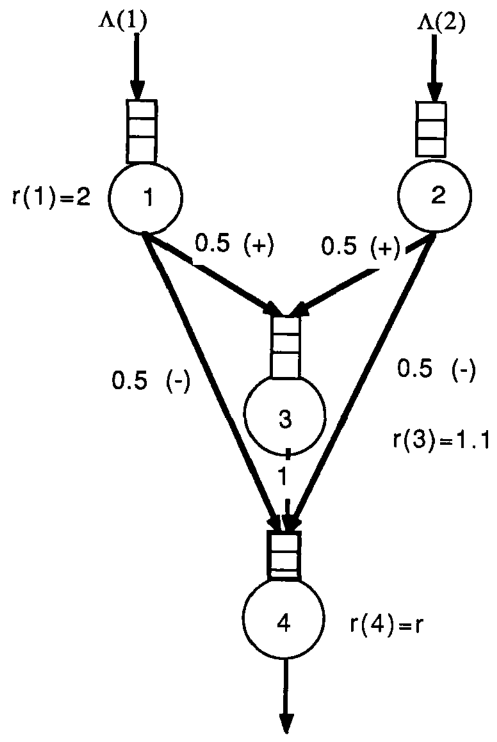

Figure 2: The random network analog of the formal neural network shown in Figure 1; here $p^+(1,3) = p^+(2,3) = 0.5$, $p^+(3,4) = 1$, and $p^-(1,4) = p^-(2,4) = 0.5$.

cumulate at neurons, and are then sent off to other neurons according to a probabilistic transition rule as the neuron fires. Neuron firing times are exponentially distributed random variables that may differ from one neuron to another. Negative signals cancel an existing positive signal at a neuron, if one exists; otherwise they have no effect. Positive signals leaving a neuron can enter another neuron as a positive or negative signal, or simply leave the system. System departure can therefore be used to simulate output signals, as well as to represent a "loss" of neuron potential. Poisson external arrivals of positive and negative signals occur. We show that this network has a product form that leads to a compact and elegant representation for its steady-state behavior. We prove that the nonlinear signal flow equations of the model have a unique solution when the network has a feedforward structure. An analogy with the usual formal neural network model is shown.

Appendix A Proof of Theorem 1

Since $\{k(t) : t > 0\}$ is a continuous time Markov chain, it satisfies the usual Chapman–Kolmogorov equations; thus, in steady state it can be seen that $p(k)$ satisfies the following global balance equations:

$$\begin{aligned} p(k)\sum_i[\Lambda(i) &+ (\lambda(i)+r(i))\mathbf{1}[k_i>0]] \\ &= \sum_i[p(k_i^+)r(i)d(i) + p(k_i^-)\Lambda(i)\mathbf{1}[k_i>0] + p(k_i^+)\lambda(i) \\ &\quad + \sum_j(p(k_{ij}^{\pm})r(i)p^+(i,j)\mathbf{1}[k_j>0] \\ &\quad + p(k_i^+)r(i)p^-(i,j)\mathbf{1}[kj=0] + p(k_{ij}^{++})r(i)p^-(i,j))] \end{aligned} \tag{A.1}$$

where the vectors used are defined by

$$\begin{aligned} k_i^+ &= (k_1,\ldots,k_i+1,\ldots,k_n) \\ k_i^- &= (k_1,\ldots,k_i-1,\ldots,k_n) \\ k_{ij}^{\pm} &= (k_1,\ldots,k_i+1,\ldots,k_j-1,\ldots,k_n) \\ k_{ij}^{++} &= (k_1,\ldots,k_i+1,\ldots,k_j+1,\ldots,k_n) \end{aligned}$$

and $\mathbf{1}[X]$ is the characteristic function that takes the value 1 if X is true and 0 otherwise. These vectors have no meaning whenever any of their elements are negative; in such cases, the corresponding probabilities in the global balance equation are understood to be zero.

We now verify that the product from equation 2.3 satisfies the equation. Substituting 2.3 in equation A.1 we have:

$$\begin{aligned} &\sum_i[\lambda(i) + (\lambda(i)+r(i))\mathbf{1}[k_i>0]] \\ &= \sum_i[q_ir(i)d(i) + \Lambda(i)\mathbf{1}[k_i>0]/q_i + \lambda(i)q_i \\ &\quad + \sum_j(q_ir(i)p^+(i,j)\mathbf{1}[k_j>0]/q_j \\ &\quad + q_ir(i)p^-(i,j)\mathbf{1}[k_j=0] + q_iq_jr(i)p^-(i,j))] \end{aligned} \tag{A.2}$$

Using equations 2.3 and 2.1 we simplify the equation further in several steps:

$$\begin{aligned} &\sum_i[\Lambda(i) + (r(i)+\lambda(i))\mathbf{1}[k_i>0]] \\ &= \sum_i[q_ir(i)d(i) + \Lambda(i)\mathbf{1}[k_i>0]/q_i + \lambda(i)q_i] \\ &\quad + \sum_j\{(\lambda^+(j)-\Lambda(j))\mathbf{1}[k_j>0]/q_j + (\lambda^-(j)-\lambda(j))1[k_j=0] \end{aligned} \tag{A.3}$$

$$
\begin{aligned}
&+ (\lambda^-(j) - \lambda(j))q_j\} \\
&= \sum_i [q_i r(i) d(i) + (\lambda^-(i) - \lambda(i))\mathbf{1}[k_i = 0] \\
&+ \lambda^+(i)\mathbf{1}[k_i > 0]/q_i + \lambda^-(i) q_i] \\
&= \sum_i [q_i r(i) - \sum_j q_i r(i) p(i,j) + (\lambda^-(i) - \lambda(i))\mathbf{1}[k_i = 0] \\
&+ (r(i) + \lambda^-(i))\mathbf{1}[k_i > 0] + \lambda^-(i) q_i]
\end{aligned}
$$

Substituting equation 2.2 again, and canceling terms we remain with

$$\sum_i \Lambda(i) = \sum_i [q_i r(i) - \lambda^+(i) + \Lambda(i) + \lambda^-(i) q_i] \tag{A.4}$$

which, after using equation 2.1 becomes, for each i,

$$0 = -\lambda^+(i) + (r(i) + \lambda^-(i))\lambda^+(i)/[r(i) + \lambda^-(i)] = 0 \tag{A.5}$$

Thus, the product form is verified since equation 2.3 satisfies the global balance equations A.1. QED

References

Rumelhart, D. E., McClelland, J. L., and the PDP Research Group. 1986. *Parallel Distributed Processing,* Vols. I and II. Bradford Books and MIT Press, Cambridge, MA.

Gelenbe, E., and Mitrani, I. 1980. *Analysis and Synthesis of Computer Systems.* Academic Press, London.

Gelenbe, E., and Pujolle, G. 1986. *Introduction to Networks of Queues.* Wiley, New York.

Kandel, E. C., and Schwartz, J. H. 1985. *Principles of Neural Science.* Elsevier, Amsterdam.

Received 17 July 1989; accepted 17 August 1989.