The statistical distribution of Boolean gates in two-inputs, one-output multilayered neural networks

1990 J. Phys. A: Math. Gen. 23 3061

(http://iopscience.iop.org/0305-4470/23/13/040)

J. Phys. A: Math. Gen. **23** (1990) 3061-3072. Printed in the UK

The statistical distribution of Boolean gates in two-inputs, one-output multilayered neural networks

Mirta B Gordon and Pierre Peretto

Département de Recherche Fondamentale/SPh, Centre d'Etudes Nucléaires, 85X, 38041 Grenoble Cedex, France

Received 26 October 1989

Abstract. We study the probability of implementing Boolean functions in layered neural networks, as a function of the number of hidden units and layers. We show how these probabilities depend on the values allowed to the thresholds, and how they evolve as a function of the number of hidden layers. For two input variables, it is shown that the probability of implementing the EXCLUSIVE OR with one hidden layer remains low, even with a large number of hidden units. In the limit of an infinite number of layers, all the functions become equally probable, but probabilities already reach the asymptotic value within 10% with only five hidden layers.

1. Introduction

Neural networks are basically long-range Ising spin-glass models. A formal neuron i may be either active, $\sigma_i = +1$, or silent, $\sigma_i = -1$. Its state is determined by the 'local field' produced on it by the other neurons, through the synaptic efficacies J_{ij}. These efficacies are equivalent to magnetic exchange interactions. Once they are given, the problem in statistical mechanics is to find the equilibrium properties of the system. In contrast, the main concern in the case of neural networks is to find a complete set of interactions such that the network correctly performs a given task. Very generally, a task is a mapping between the states of a given set of neurons, the inputs, and the states of the set of output neurons. Input and output neurons may coincide: in that case the task is to associate the different metastable final states to the possible initial states of the network, with a given prescription for the dynamical updating of individual neurons. These so-called associative memories have been extensively studied with the methods developed in spin-glass theory. In order to achieve more involved tasks, like classification, generalisation, etc, more complicated architectures, with hidden neurons, have been proposed. Hidden units are relevant only for the dynamics, i.e. for the evolution of the network from its initial to its final state. In this context, input and output neurons are called 'visible' units.

Given a task and the network's connectivity, learning is the process of searching adequate synaptic strengths. Learning in systems without hidden neurons is rather well understood. When the system contains hidden units, their states are not only defined by the desired input-output mapping, but also by the updating dynamics. Finding the synaptic strengths becomes very difficult: learning in general highly interconnected networks with hidden neurons still remains an open problem. Some insight can be obtained by studying simpler, feed-forward, layered networks. In this architecture, neurons in the input layer are connected to neurons in the first hidden layer,

0305-4470/90/133061+12$03.50

which are themselves connected to those in the secod layer, and so on. Updating proceeds layer by layer, determining the states of the hidden units downstream from input to output. The fact that neurons in a hidden layer cannot influence the state of upstream layers' neurons, makes the problem tractable.

Several learning paradigms have been proposed.

(a) *Error correction paradigm* (*back-propagation algorithm*). The input layer is initialised with the patterns to be learnt. The signals are propagated from input to output units. The actual outputs are compared to the desired outputs, and the back-propagated error signals are used to modify the set of connections in a way that reduces the errors (Le Cun 1985, Rumelhart *et al* 1986).

(b) *Internal states reshuffling* (*learning by choice of internal representations*). The internal states are iteratively modified so as to yield the desired output. A perceptron algorithm is used at every step of the iteration (Meir and Domany 1988, Grossman 1989).

(c) *Building the network (the tiling algorithm).* Given a set of input-output relations, the network is constructed layer after layer. The central idea is that of faithful representation: the internal representations of inputs yielding different outputs should be different (Mezard and Nadal 1989).

(d) *Darwinian learning* (*learning by selection*). This approach is more biologically minded. Initially, connections are supposed to develop at random. Afterwards, learning has to select the adequate pathways or connectivity (Toulouse *et al* 1986, Peretto 1989).

This article deals with the last paradigm. We assume that the role of epigenesis is to make random connections between the layers of the network (sprouting phase), and the role of learning is to select the right circuits (pruning phase). The problem is that layered networks with random connections do not implement all possible functions with equal probability. Moreover, some functions may be not implementable at all by a given network. In that case, the system will be unable to react properly to stimuli whose response would call for one of these functions. Therefore, the possible behaviours of a system strongly depend on the probability distribution of Boolean functions it may implement, i.e. which mappings of binary input states onto binary output states is it able to realise when the synaptic strengths are chosen at random. On the other hand, as pointed out by Carnevali and Patarnello (1987) and Solla (1988), the fact that a given architecture may implement more than one function introduces an entropy associated with the network. Learning a given function consists of eliminating this prior entropy by adequately modifying the synaptic weights, a process usually called training, so that only the desired input-output mappings are implemented.

In this paper we present computer simulations and rigorous analytical results on the probabilities of implementing one of the simpler Boolean function, namely two-inputs, one-output, in feed-forward neural networks with random couplings. We consider networks with a variable number of hidden layers, and a variable number of hidden units per layer. The analytical results are derived in the thermodynamic limit of an infinite number of units in each hidden layer. The restriction to networks with only one output unit does not introduce any loss of generality, since an architecture with many units at the output layer may be considered as the concatenation of networks with a single output unit. The restriction to two units in the input layer is more severe. This choice has been dictated by the difficulty in handling the explosive increase of possible Boolean functions as the number of input units increases. It must be stressed, however, that the technique presented in this paper is, in principle, applicable to more general situations.

The paper is organised as follows. In the next section we present results without hidden layers, in section 3 are given the numerical results with one hidden layer, and analytic calculations are discussed in section 4. Section 5 presents the conclusions. Detailed calculations are left to appendices 1 and 2.

2. Boolean functions without hidden layers

Although implementation of Boolean functions in two-layer networks, without hidden units, is well understood (Minsky and Papert 1969), we shall consider it here in order to present the notation, the symmetries of the problem, and also the effect of the thresholds on the probability of implementing the different Boolean functions.

We consider N^0 binary input units which can be in states $\sigma_i = 1$ or $\sigma_i = -1$, $1 \leqslant i \leqslant N^0$. These units are directly coupled to the output neuron with synaptic strengths J_i. As usual, we introduce a neuron 0 clamped in the state $\sigma_0 = 1$, which acts as a threshold of value J_0. There are $S = 2^{N^0}$ possible input states, labelled s. Given an input state, and the synaptic couplings, the output neuron's state is given by the usual relation

$$\sigma^{\text{out}}(s) = \text{sgn}\left(\sum_{j=0}^{N^0} J_j \sigma_j(s) \right) \qquad 1 \leqslant s \leqslant S. \tag{1}$$

In general, there are $B = 2^S$ boolean functions of N^0 inputs. Each function τ_n, $1 \leqslant n \leqslant B$, is characterised by the S outputs $\tau_n = (\tau_n(1), \ldots, \tau_n(S))$ corresponding to each of the S possible inputs. The network implements a given function τ_n if the following S inequalities are simultaneously satisfied:

$$\tau_n(s) \sum_{j=0}^{N^0} J_j \sigma_j(s) > 0 \qquad 1 \leqslant s \leqslant S \tag{2}$$

because in that case, the output state, given by (1), will be exactly $\tau_n(s)$. Note that in general one has $S = 2^{N^0}$ inequalities and $N^0 + 1$ synaptic couplings. Because $S \geqslant N^0 + 1$, it is obvious that not always will a set of synaptic couplings $J = \{J_j\}$ satisfying (2) exist, which means that networks without hidden layers are unable to implement all the B possible Boolean functions of their inputs.

Let us concentrate in the case $N^0 = 2$: we have $B = 16$ Boolean functions, and we label the $S = 4$ possible input states as follows: state $s = 1$ corresponds to the input $(\sigma_1 = 1, \sigma_2 = 1)$, $s = 2$ to $(1, -1)$, $s = 3$ to $(-1, -1)$, and $s = 4$ to $(-1, 1)$. Equations (2) can be written explicitly

$$\tau_n(1)(J_0 + J_1 + J_2) = \lambda_1 > 0 \tag{2a}$$

$$\tau_n(2)(J_0 + J_1 - J_2) = \lambda_2 > 0 \tag{2b}$$

$$\tau_n(3)(J_0 - J_1 - J_2) = \lambda_3 > 0 \tag{2c}$$

$$\tau_n(4)(J_0 - J_1 + J_2) = \lambda_4 > 0. \tag{2d}$$

As already stated, these equations are not linearly independent: if for example the first three of them are obeyed, then constraints must be placed upon $\tau_n(4)$ to satisfy the last one. Combining (2a)-(2d) one has, dropping the subscripts n,

$$\tau(1)\lambda_1 - \tau(2)\lambda_2 + \tau(3)\lambda_3 = \tau(4)\lambda_4 \tag{3}$$

so that, because the λ_i are positive, functions with $\tau(1) = \tau(3) = -\tau(2) = -\tau(4)$, are not compatible with (3). They are precisely the exclusive or—XOR—and equivalence—EQUIV—, both well known to be not implementable without hidden units.

As pointed out by Ferran and Perazzo 1989, if the probability distribution of the synaptic strengths is the same for all neurons, excluding the threshold, then the structure of equations (2) present symmetries under permutations of the label of the input units and corresponding exchanges in the synaptic couplings. These symmetries arise because of the arbitrariness in labelling the neurons of the input layer. In addition, if the probability distributions of synaptic couplings are even, new symmetries appear in equations (2). As a consequence, the set of all the possible Boolean functions is partitioned into subsets or families. If a function τ_n is implemented by the network, then applying these symmetries generates the family of functions to which τ_n belongs. These families have no mutual overlaps, and all the functions of the same family have the same probability of being implemented by the network.

For $N^0 = 2$ it is possible to determine all the families. Let us assume that $\tau = (\tau(1), \tau(2), \tau(3), \tau(4))$ is the function implemented when the couplings take the values (J_0, J_1, J_2). Then, the outputs corresponding to states $\sigma' = (\sigma_2, \sigma_1)$, obtained by permutation of the input neurons' label, with exchanged couplings (J_0, J_2, J_1), i.e. the function $\tau' = (\tau(1), \tau(4), \tau(3), \tau(2))$, has the same probability, because it satisfies exactly the same inequalities (2). In addition to this exchange symmetry, if the probabilities of J_1 and J_2 are symmetric, the outputs to $\sigma'' = (-\sigma_1, \sigma_2)$ with couplings $(J_0, -J_1, J_2)$, namely $\tau'' = (\tau(4), \tau(3), \tau(2), \tau(1))$, also have the same probability because changing σ_1 into $-\sigma_1$ together with J_1 into $-J_1$ leaves system (2) invariant. It is easy to check that some of these symmetries are redundant. An exhaustive exploration of all the possible symmetries for $N^0 = 2$ shows that there are four families of Boolean functions, which we denote *V*, *D*, *A* and *X*. A typical member of *V* is the function whose output is always $\tau = 1$ whatever the input is. A typical member of *D* is the function $\tau = \sigma_1$, typical members of *A* are the AND $\tau = (\sigma_1 \& \sigma_2)$ and the implication $\tau = (\sigma_1 \Rightarrow \sigma_2)$. The XOR and the equivalence $\tau = (\sigma_1 \equiv \sigma_2)$, which have been shown to be not implementable, both belong to the same family, which we denote *X*. All the families are listed in table 1.

If the couplings $J = (J_0, J_1, J_2)$ are random, with probability distribution $p(J)\,\mathrm{d}J$, the probability that a function τ_n is implemented by the network without hidden units is given by

$$P_n = \int \left[\prod_{s=1}^{S} \Theta\left(\tau_n(s) \sum_{j=0}^{N^0} J_j \sigma_j(s) \right) \right] p(J)\,\mathrm{d}J = \int_{\mathscr{D}_n} p(J)\,\mathrm{d}J \tag{4}$$

where Θ is the Heaviside function, defined as usual by $\Theta(x) = 1$ if $x \geqslant 0$, and $\Theta(x) = 0$ if $x < 0$. $\mathscr{D}_n$ is the domain of the space of couplings that satisfies (2). Because of the

Table 1. The four families of Boolean functions of two inputs.

Input		Output families															
		V		*D*				*A*								*X*	
σ_1	σ_2	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7	τ_8	τ_9	τ_{10}	τ_{11}	τ_{12}	τ_{13}	τ_{14}	τ_{15}	τ_{16}
+	+	+	−	+	−	+	−	+	−	−	+	−	+	−	+	+	−
+	−	+	−	+	−	−	+	−	+	+	−	−	+	−	+	−	+
−	−	+	−	−	+	−	+	−	+	−	+	+	−	−	+	+	−
−	+	+	−	−	+	+	−	−	+	−	+	−	+	+	−	−	+

mentioned symmetry of equations (2), and assuming $p(J)=p(-J)$, one needs to calculate only one representative probability for each family.

We first consider the case where the probabilities of the couplings are uniform:

$$p(J_0)=\frac{1}{2T} \qquad -T\leqslant J_0\leqslant T \tag{5a}$$

$$p(J_1)=\frac{1}{2J} \qquad -J\leqslant J_1\leqslant J \tag{5b}$$

$$p(J_2)=\frac{1}{2J} \qquad -J\leqslant J_2\leqslant J. \tag{5c}$$

We allow the thresholds J_0 to vary between bounds different than the synaptic couplings. Introducing (5) into (4) gives the following results:

$$P_V^0=\tfrac{1}{12}r^2 \tag{6a}$$

$$P_D^0=\tfrac{1}{4}(1-r)+\tfrac{1}{12}r^2 \tag{6b}$$

$$P_A^0=\tfrac{1}{8}r-\tfrac{1}{16}r^2 \tag{6c}$$

$$P_X^0=0 \tag{6d}$$

for $r=T/J\leqslant 1$; and

$$P_V^0=\frac{5}{12}-\frac{1}{3r} \tag{7a}$$

$$P_D^0=\frac{1}{12r} \tag{7b}$$

$$P_A^0=\frac{1}{48}+\frac{1}{24r} \tag{7c}$$

$$P_X^0=0 \tag{7d}$$

for $r\geqslant 1$. In particular, for $r=1$ we find: $P_V^0=P_D^0=\frac{1}{12}=0.0833$, $P_A^0=\frac{1}{16}=0.0625$. The results are represented on figure 1. Functions of families A and V, which cannot be

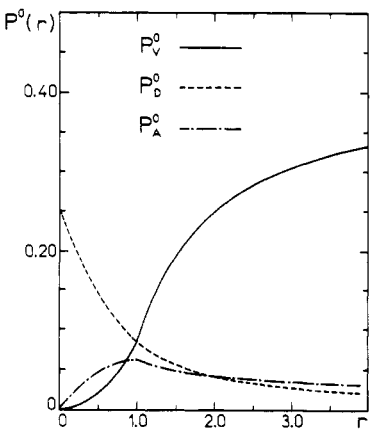

Figure 1. Probability of implementing the different families of Boolean functions without hidden units, as a function of $r=T/J$ (see text).

implemented without thresholds, have finite asymptotic probabilities for $r \to \infty$, whereas the only family implementable without thresholds, family D, has vanishing probability for $r \to \infty$.

If the synaptic couplings have identical probability densities, to be compared to the case $r = 1$, but Gaussian instead of uniform, we find slightly different values: $P_V^0 = P_D^0 = 0.108\,17$, $P_A^0 = 0.043\,87$, and obviously $P_X^0 = 0$.

3. Numerical results with one hidden layer

Suppose that we add, to the simple network of the preceding section, a certain number of hidden layers. Each layer h has N^h units, each unit i being coupled to the units j of the preceding layer by synaptic couplings J_{ij}^{h-1}, $0 \leqslant j \leqslant N^{h-1}$, which we assume are random variables. The network architecture is represented in figure 2. As in the input layer, in principle a threshold unit clamped to 1 may be introduced. However, if the number of hidden units is large enough to make sure that at least one unit implements one of the functions of family V, the threshold is not needed because functions of V type give a constant output, independent of the input state, playing therefore the same role as a threshold. This has been confirmed by computer simulations, displayed in figures 3(a) and 3(b), which represent the probabilities of implementing the four families of Boolean functions with one hidden layer, as a function of the number of hidden units, with and without threshold in the hidden layer. The couplings and thresholds have been taken at random with the same uniform distribution, so that $r = T/J = 1$ in these simulations. It is clear that for $N^1 \geqslant 30$ there is no need of a threshold hidden unit.

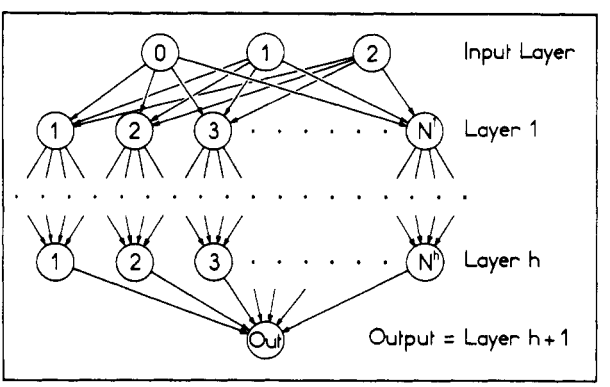

Figure 2. Network architecture with hidden layers.

Results are not very sensitive to the actual probability distribution law of the synaptic couplings: three different laws—uniform probability within an interval, Gaussian probability and random clipped synapses (in which $J_{ij}^h = 1$ or $J_{ij}^h = -1$ with probability $\frac{1}{2}$)—give similar probabilities of implementing the different Boolean functions, as is shown in figures 3 and 4.

Although it is well known that one hidden layer is enough to implement Boolean functions of family X—XOR and EQUIV—they are still 'harder' to learn than the others, even with a large number of hidden units. As we show analytically in the next section,

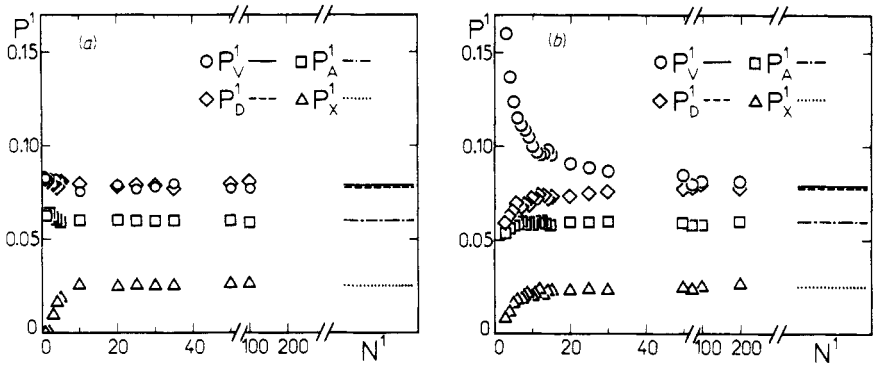

Figure 3. Probability of the different families of Boolean functions with one hidden layer, against the number of hidden units, for uniformly distributed couplings. (*a*) With a threshold unit in the hidden layer, with $r = T/J = 1$. (*b*) Without threshold unit in the hidden layer. Horizontal lines represent the asymptotic values, calculated analytically.

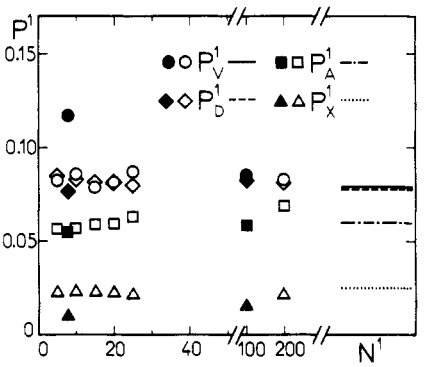

Figure 4. Probability of the different families of Boolean functions with one hidden layer, for different probability distributions of the synaptic couplings. Open symbols: clipped synapses without hidden threshold; full symbols: Gaussian distribution of synaptic strengths, with hidden threshold with the same distribution.

in the limit of an infinite number of hidden units, the volume of the space of interactions that implements these functions remains relatively small.

4. Analytic results with one and more hidden layers

Given a state s at the input layer, the state of a neuron in layer $h+1$ is a function of the states $\sigma_j^h(s)$ of units in layer h:

$$\sigma^{h+1}(s) = \operatorname{sgn}\left(\frac{1}{\sqrt{N^h}} \sum_{j=0}^{N^h} J_{ij}^h \sigma_j^h(s)\right) \qquad 1 \leqslant s \leqslant S. \tag{8}$$

The scaling factor $\sqrt{N^h}$ has been introduced to further simplify notation. The states $\sigma_j^h(s)$ are a function of states of neurons in layer $h-1$, which in turn depend on states of neurons in layer $h-2$, and so on.

We address the following question: if all the synaptic couplings J_{ij}^h are randomly chosen, what is the probability that a given unit of layer $h+1$ will realise a given Boolean function τ_n of the input states?

Adding up together the synaptic couplings of neurons of layer h that implement the same Boolean function, equations (8) can be rewritten as follows:

$$\sigma_i^{h+1}(s) = \mathrm{sgn}\left(\sum_{n=1}^{B} K_n^h \tau_n(s) \right) \tag{9}$$

where

$$K_n^h = \frac{1}{\sqrt{N^h}} \sum_{j/\sigma_j^h = \tau_n} J_{ij}^h. \tag{10}$$

Writing equation (9), we formally replaced the N^h neurons of layer h by B Boolean gates, each performing a different function τ_n, coupled to the unit we are looking at by B effective synaptic couplings, K_n^h. We assume that N^h, the number of units in layer h, is large enough to contain neurons implementing all the possible Boolean functions of the inputs. Moreover, the fraction of units performing each function is close, by the law of large numbers, to the probability of implementing the function. Therefore, if the J_{ij}^h are random variables of zero mean and variance Δ, the probability density of the effective couplings K_n can be approximated, see appendix 1, by

$$p(\{K_n^h\}) \equiv p(\{K_n^h = K_n\}) = \prod_{n=1}^{B} \frac{1}{\sqrt{2\pi\Delta^2 P_n^{h-1}}} \exp\left(-\frac{K_n^2}{2\Delta^2 P_n^{h-1}} \right). \tag{11}$$

Each sum K_n^h therefore has a Gaussian distribution, of width proportional to the probability P_n^{h-1} that the nth Boolean function is realised by the preceding layer, i.e. by a network with $h-1$ hidden layers. The probability that a neuron on layer $h+1$ (i.e. with h hidden layers) implements the Boolean function τ_m is then:

$$P_m^h \equiv P(\sigma^{h+1} = \tau_m) = \sum_{\{K_n\}\le n\le B)} p(\{K_n^h\})\, p(\tau_m/\{K_n^h\}) \tag{12}$$

where $p(\tau_m/\{K_n^h\})$, the probability of implementing τ_m given the values $\{K_n^h\}$, can be expressed as follows:

$$p(\tau_m/\{K_n\}) = \prod_{s=1}^{S} \Theta\left\{ \tau_{ms} \sum_{n=1}^{B} K_n \tau_{ns} \right\}. \tag{13}$$

From now on we write $\tau_{ns} \equiv \tau_n(s)$.

By introducing (11) and (13) into (12), together with the integral representation of the Heaviside Θ function, one can integrate over the K_n^h. Details of the calculation are left to appendix 2. One finally gets

$$P_m^h = \frac{\sqrt{b_x^{h-1} b_y^{h-1}}}{\pi} \int_{\mathcal{L}_m} \mathrm{d}x\, \mathrm{d}y \frac{\mathrm{d}z_1\, \mathrm{d}z_2}{\pi} \exp[-(z_1^2 + z_2^2) - (b_x^{h-1} x^2 + b_y^{h-1} y^2)] \tag{14}$$

where

$$b_x^{h-1} = 2\frac{\mu_1^{h-1}}{\mu_3^{h-1}} \qquad b_y^{h-1} = 2\frac{\mu_2^{h-1}}{\mu_4^{h-1}} \tag{15}$$

are functions of the probabilities of implementing the Boolean functions with $h-1$ hidden layers:

$$\begin{aligned}\mu_1^{h-1} &= \mu_2^{h-1} = 8(P_D^{h-1} + P_A^{h-1}) \\ \mu_3^{h-1} &= 8(P_X^{h-1} + P_A^{h-1}) \\ \mu_4^{h-1} &= 8(P_V^{h-1} + P_A^{h-1}).\end{aligned} \tag{16}$$

As in the case without hidden layers, the results are independent of Δ, the dispersion in the values of the random synaptic couplings. The domains $\mathscr{D}_m$ are given by

$$\begin{aligned}&\tau_m(1)(z_1+y+x) \geqslant 0 \qquad &&\tau_m(2)(z_2+y-x) \geqslant 0 \\ &\tau_m(3)(-z_1+y+x) \geqslant 0 \qquad &&\tau_m(4)(-z_2+y-x) \geqslant 0.\end{aligned} \tag{17}$$

Introducing in (15) the results of section 2 with $r=1$, corresponding to the same dispersion for thresholds as for synaptic couplings, we find the following probabilities of implementing the different families of boolean functions with one hidden layer:

$$P_V^1 = 0.0783 \qquad P_D^1 = 0.0786 \qquad P_A^1 = 0.0597 \qquad P_X^1 = 0.0253. \tag{18}$$

These results are in good agreement with the numerical simulations presented in figures 3.

Feeding back into (14) the results (18), it is possible to calculate the probabilities of implementing Boolean functions with two hidden layers, etc. Instead of doing so, introducing the expressions deduced from (14) for the four classes of functions, into (16), and integrating out the variables z_1 and z_2, we find

$$\begin{aligned}\mu_1^h &= \mu_2^h = 2 - I_+(b_x^{h-1}, b_y^{h-1}) \\ \mu_3^h &= I_+(b_x^{h-1}, b_y^{h-1}) - I_-(b_x^{h-1}, b_y^{h-1}) \\ \mu_4^h &= I_+(b_x^{h-1}, b_y^{h-1}) + I_-(b_x^{h-1}, b_y^{h-1})\end{aligned} \tag{18a}$$

where

$$I_+(a,b) = \frac{4\sqrt{ab}}{\pi}\int_0^\infty \mathrm{d}y \int_0^y \mathrm{d}x[\exp(-ax^2-by^2)+\exp(-ay^2-bx^2)] \times(\operatorname{erf}(y+x)+\operatorname{erf}(y-x)) \tag{18b}$$

$$I_-(a,b) = \frac{8\sqrt{ab}}{\pi}\int_0^\infty \mathrm{d}y \int_0^y \mathrm{d}x[\exp(-ax^2-by^2)-\exp(-ay^2-bx^2)] \times \operatorname{erf}(y+x)\operatorname{erf}(y-x). \tag{18c}$$

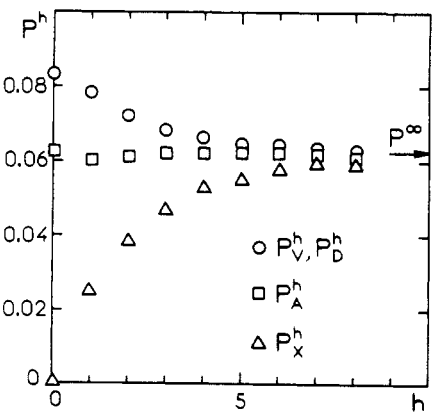

Figure 5. Probabilities against the number of hidden layers (numerical iteration of (18)).

Equations (18) have a fixed point at $b_x = b_y = 2$. At this point, $I_- = 0$ and it is easy to show that $I_+ = 1$. Moreover, from (18*a*) and (16) one gets $P_D = P_X = P_V$, and from the normalisation of probabilities, taking into account the degeneracy of each family of functions: $2P_V + 8P_A + 4P_D + 2P_X = 1$, we get $P_A = \frac{1}{8} - P_V$. Integration of (14) with $b_x = b_y = 2$ gives $P_V = \frac{1}{16}$. Therefore the fixed point corresponds to all the functions having the same probability, e.g. $\frac{1}{16}$. Indeed, numerical iteration of (18) shows that this fixed point corresponds to $h \to \infty$, convergence to this value being already achieved, within 10%, after five iterations, as is displayed in figure 5.

5. Conclusion

Let us summarise the results we arrived at. We give exact numbers regarding the volumes in the space of interactions which specifically implement the 16 possible Boolean functions mapping two binary inputs onto one binary output. This calculation of the entropies of the various functions is done in the spirit of Gardner's approach to learning (Gardner 1987), although the techniques used here slightly differ from those she introduced.

We observe that the functions are unevenly distributed when the number of layers is small. This is not really surprising since one knows for example that it is not possible to implement the XOR function in systems devoid of hidden units. When the system has one hidden layer, which is a much studied architecture, the asymptotic distribution

$$P_V^1 = 0.0783 \qquad P_D^1 = 0.0786 \qquad P_A^1 = 0.0597 \qquad P_X^1 = 0.0253$$

is already obtained for a number of units of about 30. When the number of layers increases, the distribution of Boolean function becomes uniform: all the functions have the same probability. Convergence is attained for as low a number of layers as 4 or 5. It is not clear if such a rapid convergence would be achieved for more complicated Boolean functions.

It is tempting to compare these results with experiments on human performance in learning the Boolean functions of two variables. Excluding our family V which they did not investigate, Thorpe *et al* (1988) reported that Boolean functions fell into three main groups with respect to difficulty in learning. These groups correspond to our families, and our calculated probabilities are in correpondence with the reported easiness in learning: functions in D required less trials to be learnt than functions in A, the hardest task being learning of functions in X. If there were a correspondence between our formal layers and layers in the human brain, our results would suggest that very few layers are active in learning Boolean functions, because otherwise the XOR and the EQUIV should be learnt with the same ease as the other simpler Boolean functions.

Acknowledgments

It is a pleasure to thank Michel Kerszberg, who stayed three months as a visitor to our Department at the beginning of this work, for his suggestions and for the stimulating discussions we had together.

Appendix 1

In this appendix we detail some of the calculations of section 4.

The probability density function of the $\{K_n\}$, defined by (10) is

$$p(K_1, \ldots, K_B) = \sum_{\text{partitions } \{N_n\}} p(\{K_n\}/\{N_n\}) \cdot p(\{N_n\}) \tag{A1.1}$$

where $\{N_n\}$ represents the partitions of N^h in B subsets of N_n elements implementing the same Boolean function τ_n, and satisfying $N_1 + \ldots + N_B = N^h$. The probability of such a partition is

$$p(\{N_n\}) = \frac{N!}{N_1! \ldots N_B!} [P_1^{h-1}]^{N_1} [P_2^{h-1}]^{N_2} \ldots [P_B^{h-1}]^{N_B} \tag{A1.2}$$

where P_n^{h-1} is the probability that function τ_n is implemented by a network with $h-1$ hidden layers. If $N^h \gg B$, the multinomial distribution (A1.2) is dominated by the most probable values of the arguments: $\bar{N}_n = N^h P_n^{h-1}$. Expression (A1.1) can be approximated by

$$p(\{K_n\}) \simeq p(\{K_n\}/\{\bar{N}_n\}). \tag{A1.3}$$

Therefore the K_n are now sums of $\bar{N}_n$ random variables of zero mean and variance Δ^2/N^h, and for large N^h they have a Gaussian distribution of zero mean and variance $\Delta_n^2 = \Delta^2 P_n^{h-1}$, which gives equation (11). The only assumption made is that the number of hidden units in the layer is large enough to neglect fluctuations in N_n, and that the J_{ij}^h are random independent variables of zero mean and dispersion Δ^2.

Appendix 2

Introducing (11), (13) and the integral representation of the Θ function

$$\Theta(z) = \int_0^\infty d\lambda \int_{-\infty}^{\infty} dx \, e^{-ix(z-\lambda)} \tag{A2.1}$$

into (12), it is possible to integrate out the $\{K_n\}$, to get

$$P_m^h = \int_0^\infty \left(\prod_{s=1}^{S} d\lambda_s \right) \int_{-\infty}^{\infty} \left(\prod_{s=1}^{S} \frac{dx_s}{2\pi} \right) \exp\left(-i \sum_{s=1}^{S} x_s \lambda_s \right) \exp\left(-\frac{\Delta^2}{2} \Phi_m^{h-1} \right). \tag{A2.2}$$

The biquadratic form in the exponent is

$$\Phi_m^h = \sum_{s,t=1}^{S} x_s M_{st}^h x_t = X^T M_m^h X \tag{A2.3}$$

where $X^T = (x_1, x_2, \ldots, x_s)$, X its transposed vector, and

$$(M_m^h)_{st} = \tau_{ms} \tau_{mt} \sum_{n=1}^{B} P_n^h \tau_{ns} \tau_{nt} = \tau_{ms} \tau_{mt} \sum_{\nu = \{V, D, A, X\}} P_\nu^h (\mathscr{C}_\nu^h)_{st} \tag{A2.4}$$

is a symmetric matrix, that can be expressed in terms of matrices $\mathscr{C}_\nu^h$ which only depend on the functions τ_n belonging to family ν and the probabilities P_ν^h. The matrix M_m^h

for $N^0 = 2$ is

$$M_m^h = \begin{pmatrix} 1 & \tau_{m1}\tau_{m2}y^h & \tau_{m1}\tau_{m3}z^h & \tau_{m1}\tau_{m4}y^h \\ \tau_{m1}\tau_{m2}y^h & 1 & \tau_{m2}\tau_{m3}y^h & \tau_{m2}\tau_{m4}z^h \\ \tau_{m1}\tau_{m2}z^h & \tau_{m2}\tau_{m3}y^h & 1 & \tau_{m3}\tau_{m4}y^h \\ \tau_{m1}\tau_{m4}y^h & \tau_{m2}\tau_{m4}z^h & \tau_{m3}\tau_{m4}y^h & 1 \end{pmatrix}$$

where

$$y^h = 2(P_V^h - P_X^h) \qquad z^h = 2(P_V^h - 2P_D^h + P_X^h). \tag{A2.5}$$

Its eigenvalues are

$$\begin{aligned} \mu_1^h &= \mu_2^h = 1 - z^h \\ \mu_3^h &= 1 - 2y^h + z^h \\ \mu_4^h &= 1 + 2y^h + z^h. \end{aligned} \tag{A2.6}$$

Introduction of (A2.5) into (A2.6) gives (16). Integrating the variables x_s in (A2.2) gives

$$P_m^h = \int_0^\infty \left(\prod_{s=1}^{S} d\lambda_s \right) \exp(-\tfrac{1}{2}\Lambda^\top M_m^h \Lambda) \tag{A2.7}$$

where $\Lambda^\top = (\lambda_1, \lambda_2, \ldots, \lambda_S)$. A straightforward change of variables gives equation (14).

References

Carnevali P and Patarnello S 1987 *Europhys. Lett.* **4** 1199

Ferran E A and Perazzo R P J 1989 *Preprint* CNEA, Buenos Aires, Argentina

Gardner E 1987 *Europhys. Lett.* **4** 481

Grossman T 1989 *Preprint* Weizmann Institute of Science, Israel

Le Cun Y 1985 *Proc. Cognitiva, Paris* **85** 599

Meir R and Domany E 1988 *Phys. Rev.* A **37** 608

Mezard M and Nadal J P 1989 *J. Phys. A: Math. Gen.* **22** 2191

Minsky M and Papert S 1969 *Perceptrons* (Cambridge, MA: MIT Press) (second printing (1972))

Peretto P 1989 *Int. J. Neural Syst.* **1** 31

Rumelhart D E, Hinton G E and Williams R J 1986 *Parallel Distributed Processing* vol 1 ed D E Rumelhart and J L McClelland (Cambridge, MA: MIT Press)

Solla S 1988 *Neural Networks from Models to Applications* ed L Personnaz and G Dreyfus (Paris: IDSET) p 168

Thorpe S J, O'Regan K and Pouget A 1988 *Neural Networks from Models to Applications* ed L Personnaz and G Dreyfus (Paris: IDSET) p 12

Toulouse G, Dehaene S and Changeux J-P 1986 *Proc. Natl Acad. Sci., USA* **83** 1695