J. Phys. A: Math. Gen. 24 (1991) 5617–5638. Printed in the UK

Encoding strategies in multilayer neural networks

E Elizalde, S Gómez and A Romeo

Department of Structure and Constituents of Matter, Faculty of Physics, University of Barcelona, Diagonal 647, 08028 Barcelona, Spain

Received 4 March 1991, in final form 28 June 1991

Abstract. Neural networks capable of encoding sets of patterns are analysed. Solutions are found by theoretical treatment instead of by supervised learning. The behaviour for 2^R ($R \in N$) input units is studied and its characteristic features are discussed. The accessibilities for non-spurious patterns are calculated by analytic methods. Although thermal noise may induce wrong encoding, we show how it can rid the output of spurious sequences. Further, we compute error bounds at finite temperature.

1. Introduction

Neural networks are dynamical data structures made of large numbers of highly interconnected processing units called neurons, which can usually be in two possible states, depending on a two-valued function of the states of the other units. These networks bear a qualitative resemblance to the structures found in brains, that, as far as present knowledge can explain, consist of a great deal of *real* neurons linked to each other by means of dendrites, axons and synapses. Although understanding the brain structure is nowadays still an open problem, there are simple neural network models which account for some of the capabilities of brains, particularly pattern recognition (see the *content-addressable* or *associative memory* models in Little 1974 and Hopfield 1982, 1984, or more recent examples such as Parga and Virasoro 1986, Derrida *et al* 1987, Amit *et al* 1990, Bacci *et al* 1990, Nakamura and Nishimori 1990 and Herz *et al* 1991) and optimization of constrained problems (Hopfield and Tank 1985). Event reconstruction in particle accelerators (Denby *et al* 1990, Stimpfl-Abele and Garrido 1991) and radar signal analysis (Schempp 1984, 1989) are among the most fascinating current applications of these structures.

Layered feed-forward networks are theoretical machines historically based on Rosenblatt's *perceptron* model, in which there is a layer of *input* units whose only role is to feed input patterns into the rest of the network. Next, there are one or more intermediate layers of neurons evaluating the same kind of function of the weighted sum of inputs, which, in turn, send it forward to units in the following layer. This process goes on until the final or *output* level is reached, thus making it possible to read off the computation (see e.g. Denby 1990 for a review). In the class of networks one usually deals with there are no connections leading from a neuron to units in previous layers, nor to neurons further than the next contiguous level, i.e. every unit feeds only the ones contained in the next layer.

By *simple perceptron* one refers to networks with just two layers, an input one and an output one, but without internal units in the sense that there are no intermediate

0305-4470/91/235617+22$03.50

layers. These devices have been seen to have limitations, such as the XOR problem, which do not show up in feed-forward networks with intermediate or 'hidden' layers present. Actually, it has been proven that a network with only one hidden layer can represent any Boolean function (Minsky and Papert 1969).

One of the most general problems in multilayer neural networks is to find the connection strengths and thresholds which transform several known input patterns into their corresponding output patterns—according to a given interpretation of inputs and outputs. This is precisely the problem of *encoding* as described, for instance, in Rumelhart and McClelland (1986). The typical approach is a progressive *learning* process based on the principle of *back-propagation*, which leads to a solution by a lengthy relaxation search after a sufficiently large number of iterations. However, we will show that some solutions may be found by deciding on the synaptic connections through direct inspection of the problem. It must be taken into account that the architecture of the network is generally not given beforehand. That is the reason why we are free to adjust it as necessary. The criterion, of course, will be simplicity.

In section 2 we analyse first how a network such as that in figure 1 is capable of encoding unary input and output sets. The solution therein gives insight into the way of dealing with arbitrary input and output alphabets. However, limitations to the most general situation are found, giving rise to encoding solutions based on layered network structures different from the previous one. In section 3 we return to the initial unary-pattern three-layer network and study its behaviour when the input pattern does not belong to the already encoded input alphabet. A certain type of spurious states is found, and accessibilities of non-spurious patterns are calculated, the demonstrations of some interesting mathematical properties being relegated to appendices A and B. Finite temperature is introduced in order to get rid of the spurious sequences. In section 4 we present a summary of the main results, future lines open to research and the conclusions of this paper.

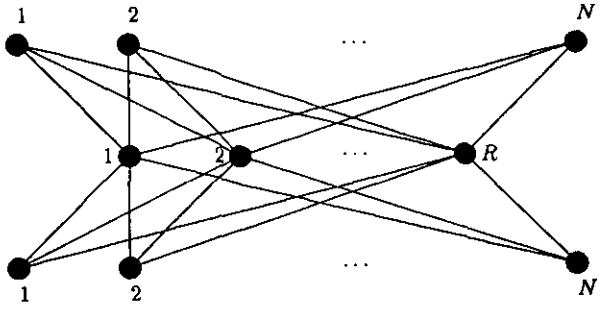

Figure 1. A network consisting of input, intermediate and output layers.

2. The encoding problem

The original problem of *encoding* is to turn p possible input patterns described by N digital units into a specified set of p patterns of M units, and to do it with the least number of intermediate processing elements. This may be seen as trying to condense all the information carried by the initial set of patterns into the tiniest space—*data*

compression—and then to recover it in the form of the corresponding output patterns—*decoding*. For the sake of simplicity we will be concerned only with the case where $N = M = p$, the reason being that, for this set-up, the association between every particular pattern and the position of each excited unit is quite easy to keep in mind.

As a technical subject, data compression can play a decisive role in the issue of encryption, as it uses many of the same principles. The idea behind this is to increase the capacity of any storage device without having to alter the actual hardware architecture, and only by an effective reduction of the storage needs of the user. Computer-based cryptography is a modern answer to the necessity of keeping sensitive data on shared systems secure, as well as a resource for data transmission, e.g. the protection of sky-to-earth station broadcasts. In addition to storage enhancement and higher security levels, the encoding of information prior to transmission saves transfer time, e.g. on phone lines.

2.1. Unary input and output sets

This is the simplest set-up, from which more involved encoding systems can be devised, as we shall show later. Let us assume an input alphabet of N symbols, each of them defined by a binary pattern of N units. The choice of unary patterns amounts to defining every element of the input set as

$$\xi^\mu \equiv (\overset{1}{-}, \ldots, \overset{\mu-1}{-}, \overset{\mu}{+}, \overset{\mu+1}{-}, \ldots, \overset{N}{-}) \qquad \mu = 1, \ldots, N \tag{2.1}$$

or, in components, $\xi_k^\mu = 2\delta_k^\mu - 1$.

We will start by requiring our network to turn a given unary input pattern of N units into an output configuration reproducing the same pattern, by means of an intermediate layer. Furthermore, for the sake of economizing on memory storage, it will be quite desirable to demand that this layer be as small as possible.

The encoding strategy to be put into practice will consist in using a hidden layer forming a binary representation of the N input characters in terms of -1 and $+1$ (instead of 0 and 1). Each element of this representation will be the binary translation of the number $\mu - 1$, associated with every pattern ξ^μ.

As a result, the dimension of this representation—in fact, the effective byte length—henceforth called R, has the following value:

$$R = \begin{cases} \log_2 N & \text{if } \log_2 N \in \mathbb{N} \\ [\log_2 N] + 1 & \text{if } \log_2 N \notin \mathbb{N}. \end{cases} \tag{2.2}$$

For instance, taking an input set of five unary patterns, one has to attach to them the numbers 0, 1, 2, 3, 4 and put them into binary form when going to the intermediate layer, which will take up only three units:

μ	ξ^μ					$\rightarrow$	σ^μ		
1	+	−	−	−	−	→	−	−	−
2	−	+	−	−	−	→	−	−	+
3	−	−	+	−	−	→	−	+	−
4	−	−	−	+	−	→	−	+	+
5	−	−	−	−	+	→	+	−	−

This simple example already shows one of the most remarkable features of this type of system, namely that the total capacity of the σ layer is not always fully exploited.

As the figure shows, the hidden layer forms just five out of eight hypothetically possible intermediate patterns. In general, a maximum of 2^R different sequences can be stored, but only when $N = 2^R$, $R \in N$ is this limit achieved.

The above translation, understood as a change of basis, may be implemented by a number of techniques on any ordinary—i.e. *non-parallel*—computer, but, since we are working on a neural network, it must be achieved by just an adequate choice of the weights or connection strengths ω_{jk} and of the threshold constants θ_j, which will relate the values of the units in both layers as follows:

$$\sigma_j = \text{sign}\left(\sum_{k=1}^{N} \omega_{jk}\xi_k - \theta_j\right) \qquad j = 1, \ldots, R. \tag{2.3}$$

The activations of the intermediate units thus defined may be written as

$$\sigma_j^\mu = (-1)^{[(\mu-1)/2^{R-j}]+1} \tag{2.4}$$

where the square brackets mean the lower integer part is taken.

Writing the expression for the unary patterns in components and making the ansatz $\omega_{jk} = \sigma_j^k$ we obtain

$$\sigma_j^\mu = \text{sign}\left(\sum_{k=1}^{N} \omega_{jk}\xi_k^\mu - \theta_j\right) = \text{sign}\left(\sum_{k=1}^{N} (-1)^{[(k-1)/2^{R-j}]+1}(2\delta_k^\mu - 1) - \theta_j\right)$$
$$= \text{sign}\left(2(-1)^{[(\mu-1)/2^{R-j}]+1} - \sum_{k=1}^{N} (-1)^{[(k-1)/2^{R-j}]+1} - \theta_j\right). \tag{2.5}$$

The equality is satisfied if

$$\theta_j = \sum_{k=1}^{N} (-1)^{[(k-1)/2^{R-j}]}. \tag{2.6}$$

Since this solution does always exist, the ansatz has been proven to work for arbitrary N.

The next step is to go from the intermediate layer to the output units. Since the output set of patterns is identical to the input one, the whole encoding process from one into the other means taking a certain ξ^μ to obtain some ξ^ν, where the index ν may be different from the given μ. If we demand that the translation be injective, the relation between the set of output indices ν and the input labels μ can be no other than a permutation of N elements. Selecting one such translation scheme amounts to making the choice of a specific permutation. It is reasonable to make a first approach to this problem by choosing the easiest element of the symmetric group, namely the identity. So, if we denote by S^μ the output pattern resulting from entering ξ^μ into the network, the set-up corresponding to the identity is that in which $S^\mu = \xi^\mu$, which, for instance, in the $N = 5$ case can be represented by

μ	σ^μ			→	S^μ				
1	−	−	−	→	+	−	−	−	−
2	−	−	+	→	−	+	−	−	−
3	−	+	−	→	−	−	+	−	−
4	−	+	+	→	−	−	−	+	−
5	+	−	−	→	−	−	−	−	+

The connection weights and thresholds must make possible the relation

$$S_i^\mu = \text{sign}\left(\sum_{j=1}^{R} \omega_{ij}\sigma_j^\mu - \theta_i\right). \tag{2.7}$$

Once more, we look for a solution which relates the weights to the values of the intermediate units:

$$\omega_{ij} = \sigma_j^i = (-1)^{[(i-1)/2^{R-j}]+1}. \tag{2.8}$$

By our assumption, we have

$$\sum_{j=1}^{R} \omega_{ij}\sigma_j^\mu = \sum_{j=1}^{R} \sigma_j^i\sigma_j^\mu \leqslant \sum_{j=1}^{R} (\sigma_j^i)^2 = R \tag{2.9}$$

i.e. since each term is a product of two signs, the weighted sum of the values of the hidden units achieves a maximum equal to R when all the pairs of signs coincide, which happens precisely for $\mu = i$. Otherwise, there must be at least one pair of opposed signs and therefore

$$\sum_{j=1}^{R} \omega_{ij}\sigma_j^\mu \leqslant R-2 \qquad \text{for } \mu \neq i. \tag{2.10}$$

Going back to (2.7), given that $S_i^\mu = \xi_i^\mu$, which has a plus sign for the unit at $i = \mu$ and minuses elsewhere, the thresholds θ_i must be such that

$$\begin{cases} \sum_{j=1}^{R} \omega_{ij}\sigma_j^\mu - \theta_i > 0 & \text{for the maximum } (\mu = i) \\ \sum_{j=1}^{R} \omega_{ij}\sigma_j^\mu - \theta_i < 0 & \text{for the rest } (\mu \neq i). \end{cases} \tag{2.11}$$

This is compatible with (2.9) and (2.10). In fact, by simply taking the thresholds within a certain range the fulfilment of these conditions is automatically ensured. This range is

$$\theta_i = R-2+\varepsilon \qquad i = 1, \ldots, N \qquad 0 < \varepsilon < 2 \tag{2.12}$$

but, in order to work with determined objects, we content ourselves with choosing

$$\theta_i = R-1 \qquad i = 1, \ldots, N. \tag{2.13}$$

For an arbitrary permutation of N elements, the picture is slightly altered to

$$\xi_k^\mu \longrightarrow \sigma_j^\mu \longrightarrow S_i = \xi_i^\nu$$

$$\begin{matrix} \omega_{jk} & \omega_{ij} \\ \theta_j & \theta_i \end{matrix}$$

$$\nu = \tau(\mu) \qquad \tau \in \{\text{permutations of } N \text{ elements}\}.$$

All these steps can be equally retraced with the only difference that the weights ω_{ij} now coincide with the σ up to a label reshuffle, i.e. instead of (2.8) we have $\omega_{\tau(\mu)j} = \sigma_j^\mu$, or, equivalently,

$$\omega_{\mu j} = \sigma_j^{\tau^{-1}(\mu)} = (-1)^{[(\tau^{-1}(\mu)-1)/2^{R-j}]+1}. \tag{2.14}$$

Thus, our general solution is

$$\begin{cases} \omega_{ij} = (-1)^{[(\tau^{-1}(i)-1)/2^{R-j}]+1} & i = 1, \ldots, N \qquad j = 1, \ldots, R \\ \theta_i = R-1 & i = 1, \ldots, N. \end{cases} \tag{2.15}$$

2.2. Arbitrary input and output sets

The obvious continuation of the work so far is an enhancement of the system described above so as to make it capable of translating binary patterns of a given arbitrary input set into elements of another arbitrary—but also specified—output set. The arbitrariness at the output level allows the encoding to be non-injective, i.e. there may be $\mu_1 \neq \mu_2$ such that $S^{\mu_1} = S^{\mu_2}$. If ζ^μ, $\mu = 1, \ldots, N$ denotes the arbitrary input set and S^μ, $\mu = 1, \ldots, N$ are the output patterns, in general different from the ζ^μ, we will require our network to produce $S^{\tau(\mu)}$ as output whenever ζ^μ is read as input, τ being any specified permutation of N elements. Actually, the use of τ is redundant in the sense that, as there is now no natural relationship between the ordering of the input and output patterns, different τ may at any rate be interpreted as different label reshuffles added to the identity permutation in the output set. We will still assume that the number of units at the input and output levels are equal to the number of patterns.

2.2.1. Five layers. A quite simple alternative that takes advantage of the above results is the actual enlargement of our unary pattern permuter system, by turning the old input and output layers into intermediate ones and adding two further layers, where the new arbitrary sets can be read and written, as depicted in the following diagram:

$$\begin{array}{ccccccc} \zeta_l^\mu & \longrightarrow & \xi_k^\mu & \longrightarrow & \sigma_j^\mu & \longrightarrow & \xi_i^{\tau(\mu)} \longrightarrow S_h^{\tau(\mu)} \\ & \omega_{kl} & & \omega_{jk} & & \omega_{ij} & \omega_{hi} \\ & \theta_k & & \theta_j & & \theta_i & \theta_h \end{array}$$

We use l indices to denote each unit of the input patterns and h indices to label each neuron in the output layer. While the three intermediate levels work exactly as in the previous network, two new sets of connection weights and thresholds will have to implement the translation from arbitrary sequences to unary patterns and the other way around.

It is not difficult to guess

$$\omega_{kl} = \zeta_l^k \tag{2.16}$$

the reason for this choice being that it makes the weighted sum of the input ζ^μ achieve a maximum of value N precisely for $\mu = k$, i.e.

$$\sum_{l=1}^{N} \omega_{kl}\zeta_l^\mu = \sum_{l=1}^{N} \zeta_l^k \zeta_l^\mu \leqslant \sum_{l=1}^{N} (\zeta_l^k)^2 = N. \tag{2.17}$$

As we have seen, this type of reasoning works when we require the next layer to be in a state where one neuron is on and the others are off, which is indeed the case for the unary configurations ξ^μ. Taking this into account, a suitable choice for the threshold is

$$\theta_k = N - 1 \qquad k = 1, \ldots, N. \tag{2.18}$$

Finally, the equality that has to be satisfied for the last step is

$$\begin{aligned} S_h^{\tau(\mu)} &= \operatorname{sign}\left(\sum_{i=1}^{N} \omega_{hi}\xi_i^{\tau(\mu)} - \theta_j \right) = \operatorname{sign}\left(\sum_{i=1}^{N} \omega_{hi}(2\delta_i^{\tau(\mu)} - 1) - \theta_h \right) \\ &= \operatorname{sign}\left(2\omega_{h\tau(\mu)} - \sum_{i=1}^{N} \omega_{hi} - \theta_h \right) \end{aligned} \tag{2.19}$$

which clearly holds if $\omega_{h\tau(\mu)} = S_h^{\tau(\mu)}$, i.e.

$$\omega_{hi} = S_h^i \tag{2.20}$$

and

$$\theta_h = -\sum_{\nu=1}^{N} S_h^\nu. \tag{2.21}$$

This constitutes a solution. Even though we have retained both the type of structure and the conceptual simplicity of the first network, this design has the disadvantage that it uses up $2N+R$ intermediate units in its three intermediate layers, which can be rather wasteful as N grows large.

2.2.2. Three layers. Another option is to give up the use of the reduced layer, i.e. the one with R units. For $N = 2^R$ this substructure acts as a sieve in the sense that, even if a *non*-unary pattern reaches the previous layer, the possible states of the reduced one are such that the signals sent forward to the next layer will give rise to a unary sequence anyway. In other words, the R-bit byte works as a perfect filter. As a result of this construction, no matter whether an input pattern belongs to the set of ζ^μ or not, the corresponding output will be one of the S^μ. Nevertheless, we shall see that, as far as the input and output alphabets themselves are concerned, the same translation task can be performed by a network with just one intermediate layer of N units. Although the removal of the reduced layer may mean the loss of this sifting power, it will no doubt be a substantial gain in storage economy.

There are several possible schemes of this sort, one of them being

$$\begin{array}{ccccc} \zeta_l^\mu & \longrightarrow & \xi_k^\mu & \longrightarrow & S_h^{\tau(\mu)} \\ & \omega_{kl} & & \omega_{hk} & \\ & \theta_k & & \theta_h & \end{array}$$

Since this is almost like cutting out two layers and two sets of connections from the five-level device, the weights and thresholds for what remains are easily found to be

$$\omega_{kl} = \zeta_l^k \qquad \theta_k = N - 1 \tag{2.22}$$

and

$$\omega_{hk} = S_h^{\tau(k)} \qquad \theta_h = -\sum_{\nu=1}^{N} S_h^\nu. \tag{2.23}$$

This scheme suggests using bytes N bits long. Although good, this solution does not seem to be quite optimal, as one might wish to do the same task with a reduced intermediate level—say, a shorter byte—instead of one of N units. However, the answer we have found is a bit discouraging, and lies in the following.

Theorem 1. It is not possible to encode through the scheme

$$\begin{array}{ccccc} \zeta_l^\mu & \longrightarrow & \sigma_j^\mu & \longrightarrow & S_h^{\tau(\mu)} \\ & \omega_{jl} & & \omega_{hj} & \\ & \theta_j & & \theta_h & \end{array}$$

for arbitrary sets $\{\zeta_l^\mu\}$ and $\{S_h^{\tau(\mu)}\}$.

Proof. It suffices to show particular examples of pattern sets leading to contradiction.

(i) *Special choice of output patterns.*

μ	σ^μ				S^μ		
1	−	−	⟶	−	−	−	−
2	−	+	⟶	+	−	+	+
3	+	−	⟶	+	−	+	−
4	+	+	⟶	−	+	+	−

For this choice of the output alphabet, the first column of the S, i.e. S_1^μ, $\mu=1,2,3,4$—marked out in the table—happens to be the 'exclusive OR', or XOR, Boolean function. As has been shown in Minsky and Papert (1969) (see also Hertz and Palmer 1988 and other works), this rather elementary computation cannot be solved by a simple perceptron, which amounts to stating that the task of obtaining S_1^μ from the σ^μ can by no means be performed by a single step from the reduced layer to that containing the S^μ. Moreover, this sort of inconsistency will show up whenever we take an $N=4$, $R=2$ system where one of the output columns reproduces the values of the XOR function. For arbitrary N we would encounter the same hindrance if an output column took on the values of the generalized *parity*—or rather *oddness*—function, which is defined to be $+1$ when there is an odd number of plus signs in the input and -1 otherwise, and constitutes the high-dimensional extension of XOR.

(ii) *Special case of input patterns.*

μ	ζ^μ				⟶	σ^μ	
1	−	−	+	+	⟶	−	−
2	+	+	−	−	⟶	−	+
3	−	+	−	+	⟶	+	−
4	+	−	+	−	⟶	+	+

Making use of our freedom to select arbitrary sets of input patterns, we have picked one whose elements are not linearly independent. As a result, the contradiction does now arise from the ensuing expressions limiting the thresholds. Consideration of the relations for $\mu=1$ and $\mu=2$ leads to $\theta_1>0$ whereas the inequalities for $\mu=3$ and $\mu=4$ require $\theta_1<0$, leaving no chance of realizing this scheme. The same kind of reasoning is applicable to arbitrary N. □

2.2.3. Four layers. Even though the above theorem bans the possibility of implementing the theoretically optimal scheme, we can still hope to get close to it in some sense. The difficulty found in the step from the input to the intermediate layer will be removed by demanding that the ζ^μ, although arbitrary, be *linearly independent*. As for the way from the σ units to the output cells, we will introduce a further intermediate layer, working exactly as in the five-layer scheme, i.e.

$$\begin{array}{ccccc} \zeta_l^\mu & \longrightarrow & \sigma_j^\mu & \longrightarrow & \xi_i^{\tau(\mu)} \longrightarrow S_h^{\tau(\mu)} \\ & \omega_{jl} & & \omega_{ij} & \omega_{hi} \\ & \theta_j & & \theta_i & \theta_h \end{array}$$

where the only unknowns are the ω_{jl} and θ_j. One way of doing so is to look for two

successive affine transformations such that

$$\begin{array}{ccccc} \zeta^\mu & \longrightarrow & \xi^\mu & \longrightarrow & \sigma^\mu \\ & \xi = A\zeta + B & & \sigma = C\xi + D. & \end{array}$$

A solution satisfying this is

$$\begin{cases} A_{kl} = 2(\zeta)^{-1}_{kl} & k, l = 1, \ldots, N \\ B_k = -1 & k = 1, \ldots, N \end{cases} \tag{2.24}$$

where $(\zeta)^{-1}$ is the inverse of the matrix $(\zeta)_{l\mu} \equiv \zeta^\mu_l$, and

$$\begin{cases} C_{j\mu} = \frac{1}{2}\sigma^\mu_j & j = 1, \ldots, R \qquad \mu = 1, \ldots, N \\ D_j = \frac{1}{2}\sum_{\nu=1}^{N} \sigma^\nu_j & j = 1, \ldots, R. \end{cases} \tag{2.25}$$

Composing both maps one gets

$$\sigma = C\xi + D = C(A\zeta + B) + D = CA\zeta + CB + D \tag{2.26}$$

and, since in this case $CB + D$ turns out to be zero, the transformation becomes just a linear map, that is

$$\sigma_j = \sum_l \omega_{jl}\zeta_l \qquad \omega_{jl} = \sum_\nu \sigma^\nu_j (\zeta)^{-1}_{\nu l} = \sum_{\nu=1}^{N} (-1)^{[(\nu-1)/2^{R-j}]+1} (\zeta)^{-1}_{\nu l}. \tag{2.27}$$

Table 1. Different network structures. The type of arrow drawn indicates the sort of functions of the weighted sum minus threshold that can be alternatively used yielding the same result. A simple arrow denotes sign function; one with a tail means that the argument is twice a sign, so instead of taking the sign we can just divide by two. The double arrow means that the sign function is absolutely redundant.

(a)	ζ^μ_l	$\xrightarrow{\begin{cases}\omega_{kl}\\ \theta_k\end{cases}}$ ξ^μ_k	$\xmapsto{\begin{cases}\omega_{jk}\\ \theta_j\end{cases}}$ σ^μ_j	$\xrightarrow{\begin{cases}\omega_{ij}\\ \theta_i\end{cases}}$ $\xi^{\tau(\mu)}_i$ $\xmapsto{\begin{cases}\omega_{hi}\\ \theta_h\end{cases}}$ $S^{\tau(\mu)}_h$
(a′)	ζ^μ_l	$\xrightarrow{\begin{cases}\omega_{kl}\\ \theta_k\end{cases}}$ ξ^μ_k	$\xrightarrow{\begin{cases}\omega_{ik}\\ \eta_i\end{cases}}$ $\xi^{\tau(\mu)}_i$	$\xmapsto{\begin{cases}\omega_{hi}\\ \theta_h\end{cases}}$ $S^{\tau(\mu)}_h$
(b)	ζ^μ_l	$\xrightarrow{\begin{cases}\omega_{kl}\\ \theta_k\end{cases}}$ ξ^μ_k	$\xmapsto{\begin{cases}\omega_{hk}\\ \theta_h\end{cases}}$	$S^{\tau(\mu)}_h$
(b′)	ζ^μ_l	$\xrightarrow{\begin{cases}\omega_{il}\\ \kappa_i\end{cases}}$	$\xi^{\tau(\mu)}_i$	$\xmapsto{\begin{cases}\omega_{hi}\\ \theta_h\end{cases}}$ $S^{\tau(\mu)}_h$
(c)	ζ^μ_l	$\xRightarrow{\omega_{jl}}$ σ^μ_j	$\xrightarrow{\begin{cases}\omega_{ij}\\ \theta_i\end{cases}}$ $\xi^{\tau(\mu)}_i$	$\xmapsto{\begin{cases}\omega_{hi}\\ \theta_h\end{cases}}$ $S^{\tau(\mu)}_h$
(c′)	ζ^μ_l	$\xrightarrow{\begin{cases}\Omega_{il}\\ \theta_i\end{cases}}$	$\xi^{\tau(\mu)}_i$	$\xmapsto{\begin{cases}\omega_{hi}\\ \theta_h\end{cases}}$ $S^{\tau(\mu)}_h$

$$\begin{cases} \longrightarrow \text{sign} \\ \longmapsto \text{sign}\{\ldots\} = \frac{1}{2}\{\ldots\} \\ \Longrightarrow \text{sign}\{\ldots\} = \{\ldots\} \end{cases}$$

Table 2. Expressions for the weights and thresholds.

$R=[\log_2 N]+1-\delta_{[N]N}$	
$\xi_k^\mu = 2\delta_k^\mu - 1$ $\sigma_j^\mu = (-1)^{[(\mu-1)/2^{R-j}]+1}$	
$\omega_{kl} = \zeta_l^k$	$\theta_k = N-1$
$\omega_{jk} = \sigma_j^k$	$\theta_j = -\sum_{\nu=1}^{N} \sigma_j^\nu (0 \text{ if } N=2^R)$
$\omega_{ij} = \sigma_j^{\tau^{-1}(i)}$	$\theta_i = R-1$
$\omega_{hl} = S_h^l$	$\theta_h = -\sum_{\nu=1}^{N} S_h^\nu$
$\omega_{ik} = \frac{1}{2}\sum_{j=1}^{R} \omega_{ij}\omega_{jk}$	$\eta_i = R-1-\sum_{k=1}^{N} \omega_{ik}$
$\omega_{hk} = S_h^{\tau(k)}$	θ_h
$\omega_{il} = \zeta_l^{\tau^{-1}(i)}$	$\kappa_l = N-1$
$\omega_{jl} = \sum_{\nu=1}^{N} \sigma_j^\nu(\zeta)_{\nu l}^{-1}$	0
$\Omega_{il} = \sum_{j=1}^{R} \omega_{ij}\omega_{jl}$	θ_i

2.2.4. Further variants. In addition to the preceding ones, we have found other schemes which are, in fact, only variations of those already described. For instance, departing from the five-layer network (N, N, R, N, N), we have composed the two intermediate transformations, thus getting rid of the σ layer at the expense of using some more involved weights and thresholds, the result being an (N, N, N, N) structure called a' in the diagram. Next, we have found b' moving back the permutation τ and b from the second to the first transformation. Finally, the composition of the first two steps of c gives a three-layer network (N, N, N) called c'.

By way of summarizing and completing this picture, all the quantities occurring are listed in tables 1 and 2.

3. Accessibilities

Once an encoding scheme has been chosen, one might wonder which is the result when the input pattern is none of the input alphabet. It may seem unjustified, since different encoding solutions will produce different outputs. However, this is the basis of almost all the current applications of multilayer neural networks: first, weights and thresholds are calculated (e.g. by means of learning) and then the network is used to predict, classify or interpolate. Lots of examples may be given, such as hyphenation algorithms, protein secondary structure determiners and family tree relationship predictors (Rumelhart *et al* 1986).

In what follows we shall concern ourselves with the working of the initial unary-pattern three-layer permuter device. In fact, if the input pattern is not unary the network does not work! The reason is that the *fields*

$$h_j = \sum_{k=1}^{N} \omega_{jk}\xi_k - \theta_j \tag{3.1}$$

may vanish for some j, and then $\sigma_j = \text{sign}\, h_j$ is no longer well defined. There are several possible ways out:

1. By redefining the sign function, either as

$$\text{sign}(x) \equiv \begin{cases} -1 & \text{if } x < 0 \\ 1 & \text{if } x \geqslant 0 \end{cases}$$

   or the other way around. This, however, is a rather unpleasant solution as it brings about a manifest asymmetry between the chances of obtaining -1 and $+1$.
2. Shifting the thresholds $\theta_j \to \theta_j + \varepsilon$, $|\varepsilon| < 1$, i.e. non-integer values are now allowed. Again, we get an unwanted asymmetry, since all the zero fields would, from now on, give a certain sign depending on the target unit but not on the input pattern.
3. Making the intermediate σ_j units take on three values, -1, 0 and $+1$:

$$\text{sign}(x) \equiv \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0. \end{cases}$$

4. Introducing a finite—but low—temperature, and making the activations be stochastic. Then, the sign taken on by every unit is no longer the result of a deterministic function, but rather a random variable, for which the probabilities of obtaining -1 or $+1$ are given by sigmoid curves whose shapes depend on $\beta \equiv 1/T$ and approach that of a step function as β goes to infinity (deterministic limit). The condition that this temperature should be low is necessary in order to preserve—after taking an average over many realizations—the same result as for $T = 0$ when the input patterns are the ξ^μ.

3.1. Accessibilities of a three-valued unit intermediate layer

The third option calls for a study of the accessibility of the different σ_j. By accessibility of a binary pattern, thought of as a *memory*, we mean the fraction of starting arbitrary states which leads to that particular pattern (Hopfield *et al* 1983):

$$A(\sigma) = \frac{\text{No. of input patterns giving } \sigma}{\text{No. of possible different input patterns } (=2^N)}.$$

As happens in associative memory networks, different memories of the same size may in general not be equally easy to recall. The parallel to the appearance of *spurious* memories in an associative memory device is now the existence of the—to some extent unwanted—zero states. An open question about our zero-temperature encoding system is how to interpret the different sequences which end up in the same σ state. These sequences, rather than resembling each other in the sense of being close by Hamming distance—as happens in associative memory—are such that they tend to produce a value σ_j in the jth unit depending on the similarity between the input pattern ξ and the jth row of ω, which we shall call ω_j.

A most interesting property of our scheme is the vanishing of all the input thresholds whenever the number of external units equals an exact power of two, i.e.

$$\theta_j = \sum_{k=1}^{N} (-1)^{[(k-1)/2^{R-j}]} = -\sum_{k=1}^{N} \omega_{jk} = 0 \qquad \text{for } N = 2^R \qquad j = 1, \ldots, R \tag{3.2}$$

as can be seen by looking at the (ω) matrix, since for $N = 2^R$ the sum of all the coefficients in each row is zero.

A little thought shows that the fields h_j can take as values $-N$, $-N+2, \ldots, 0, \ldots, N-2, N$, depending on the number of sign coincidences between ξ and ω_j.

Denoting by $f(h_j)$ the frequency of h_j, or number of possibilities that the weighted sum equals the given quantity h_j, we obtain

$$f(h_j) = \binom{N}{(N-h_j)/2} \tag{3.3}$$

and therefore

$$f(h_j = 0) = \binom{N}{N/2} \qquad f(h_j \neq 0) = 2^N - \binom{N}{N/2}. \tag{3.4}$$

We shall reason below that the accessibility of every *non-spurious* pattern—i.e. free from zeros—may be put in terms of just the joint frequencies or probabilities that a number of field components vanish. It is for this reason that the calculation of these joint frequencies must be understood first. We start by considering

$$f(h_i = 0, h_j = 0) \qquad i \neq j.$$

A fundamental property of our connection weight matrix is that for this same situation, $N = 2^R$, their rows are mutually orthogonal. Since the coefficients are -1 and $+1$, this means that for any two given rows, one-half of the coefficients coincide and the other half are just opposite.

The frequency we are going to evaluate is the total number of input possibilities for the ξ, unary or not, such that the equations

$$\left.\begin{aligned} &\omega_{i1}\xi_1 + \omega_{i2}\xi_2 + \ldots + \omega_{iN}\xi_N = 0 \\ &\omega_{j1}\xi_1 + \omega_{j2}\xi_2 + \ldots + \omega_{jN}\xi_N = 0 \end{aligned}\right\} \tag{3.5}$$

are simultaneously satisfied. By the above orthogonality property, we can put

$$\begin{aligned} &\omega_{ik_1} = \omega_{jk_1}, \ldots \qquad \omega_{ik_{N/2}} = \omega_{jk_{N/2}} \\ &\omega_{ik'_1} = -\omega_{jk'_1}, \ldots \qquad \omega_{ik'_{N/2}} = -\omega_{jk'_{N/2}} \end{aligned} \tag{3.6}$$

where we have denoted by $k_1, \ldots, k_{N/2}$ the indices for which the coefficients coincide and by $k'_1, \ldots, k'_{N/2}$ those for which they are opposite. In terms of these sets of indices, the system of two equations reads

$$\begin{aligned} &\underbrace{\omega_{ik_1}\xi_{k_1} + \ldots + \omega_{ik_{N/2}}\xi_{k_{N/2}}}_{A} + \underbrace{\omega_{ik'_1}\xi_{k'_1} + \ldots + \omega_{ik'_{N/2}}\xi_{k'_{N/2}}}_{B} = 0 \\ &\omega_{ik_1}\xi_{k_1} + \ldots + \omega_{ik_{N/2}}\xi_{k_{N/2}} - \omega_{ik'_1}\xi_{k'_1} - \ldots - \omega_{ik'_{N/2}}\xi_{k'_{N/2}} = 0 \end{aligned} \tag{3.7}$$

where A and B are partial weighted sums defined as shown. The resulting system for these two new variables is immediately solved:

$$\left.\begin{aligned} A + B = 0 \\ A - B = 0 \end{aligned}\right\} \Rightarrow A = B = 0 \tag{3.8}$$

which, in turn, implies

$$\begin{cases} \omega_{ik_1}\xi_{k_1} + \ldots + \omega_{ik_{N/2}}\xi_{k_{N/2}} = 0 \\ \omega_{ik'_1}\xi_{k'_1} + \ldots + \omega_{ik'_{N/2}}\xi_{k'_{N/2}} = 0. \end{cases} \tag{3.9}$$

Now, the unknowns in each equation are independent. Thus, for each of them, we can make the same reasoning as before when $h_j = 0$, with the only difference that N has to be replaced with $N/2$, as each identity contains just a half of the original number

of terms. Thus

$$f_{N/2}(h_i=0)=\binom{N/2}{N/4} \tag{3.10}$$

and the joint frequency is found as a joint probability:

$$f_N(h_i=0,\, h_j=0)=f_{N/2}(h_i=0)f_{N/2}(h_j=0)=\binom{N/2}{N/4}^2. \tag{3.11}$$

After a lengthier calculation we have also found

$$f(h_i=0,\, h_j=0,\, h_k=0)=\sum_{\substack{A=-N/4\\ \text{step } 2}}^{N/4}\binom{N/4}{(N/4-A)/2}^4=\sum_{k=0}^{N/4}\binom{N/4}{k}^4 \qquad i\neq j\neq k\neq i. \tag{3.12}$$

The following joint frequency is a bit more difficult to compute, but it gives an idea of what has to be done for any number of vanishing field components. If we want to calculate

$$f(h_i=0,\, h_j=0,\, h_k=0,\, h_l=0) \qquad i, j, k, l, \text{ all different}$$

after writing down the equations, we pick the partial sums common to four (A), three (B, C, D, E) and two (F, G, H) of them. Then, we express the equations using these variables:

$$\left.\begin{aligned} &A+B+C+D-E+F+G+H=0\\ &A+B+C-D+E+F-G-H=0\\ &A+B-C+D+E-F+G-H=0\\ &A-B+C+D+E-F-G+H=0 \end{aligned}\right\}. \tag{3.13}$$

Next, we find the degree of indetermination—now eight unknowns minus four equations equals four degrees of freedom—in order to know how many unknowns remain arbitrary. The system will be solved by putting the rest as a function of the arbitrary ones. Considering A, B, C and D to be free, we get

$$\begin{cases} E=-2A-B-C-D\\ F=-A-B-C\\ G=-A-B-D\\ H=-A-C-D. \end{cases} \tag{3.14}$$

After considering the ranges of the free variables, we apply (3.3) with N replaced by $N/8$, as a result of which, the whole joint frequency is given by

$$\begin{aligned} &f(h_i=0,\, h_j=0,\, h_k=0,\, h_l=0)\\ &\quad=\sum_A\sum_B\sum_C\sum_D f(A)f(B)f(C)f(D)f(E(A,B,C,D))\\ &\qquad\times f(F(A,B,C,D))f(G(A,B,C,D))f(H(A,B,C,D))\\ &\quad=\sum_{a=0}^{N/8}\sum_{b=0}^{N/8}\sum_{c=0}^{N/8}\sum_{d=0}^{N/8}\binom{N/8}{a}\binom{N/8}{b}\binom{N/8}{c}\binom{N/8}{d}\\ &\qquad\times\binom{N/8}{2a+b+c+d-N/4}\binom{N/8}{N/4-(a+b+c)}\\ &\qquad\times\binom{N/8}{N/4-(a+b+d)}\binom{N/8}{N/4-(a+c+d)} \end{aligned} \tag{3.15}$$

where several index rearrangements have been performed. Up to this point, the binomial coefficients are to be understood in the general sense, i.e. when the number downstairs is negative or when the difference between upstairs and downstairs is a negative integer, they must be taken to be zero. Otherwise we would have to state explicitly that the sum is restricted to a, b, c and d yielding non-vanishing coefficients. In fact, since many terms give a zero contribution, the calculation of these sums is much easier than it looks.

The procedure described is completely general. Following these steps for any number of vanishing fields, one considers the common pieces in the initial equations, solves an undetermined linear system, uses the expressions of the frequencies for the values of weighted sums and arrives at multiple sums involving binomial coefficients only. The multiplicity of the final sum is always the degree of indetermination of the linear system.

As anticipated, we are going to find the accessibilities in terms of the preceding frequencies only, namely the $f(h_1=0,\ldots,h_j=0)$, $1\leq j\leq R$, which we shall call *orthogonalities*. We start with the total number of possible different input binary patterns, i.e. 2^N. This figure must be equal to the sum of the frequencies for all the possible sorts of field configurations for the σ level, thus

$$2^N=\sum_{j=0}^{R}\sum_{\{k_1,\ldots,k_j\}} f(h_1\neq 0,\ldots,h_{k_1}=0,\ldots,h_{k_j}=0,\ldots,h_R\neq 0) \tag{3.16}$$

where $\{k_1,\ldots,k_j\}$ denotes a choice of j indices among the R existing ones. The indices picked are those for which the associated field component vanishes, while the rest are non-zero. f denotes the corresponding rate of occurrence, i.e. the number of input patterns yielding that type of field configuration. Since j runs from 0 to R, this sum ranges over all the possibilities that can take place. It can be argued that these frequencies depend on the number of components that vanish, but not on the position they are located at, i.e.

$$f(h_1\neq 0,\ldots,h_{k_1}=0,\ldots,h_{k_j}=0,\ldots,h_R\neq 0)$$
$$=f(h_1=0,\ldots,h_j=0,h_{j+1}\neq 0,\ldots,h_R\neq 0)$$

for all possible rearrangements. Therefore

$$2^N=\sum_{j=0}^{R}\binom{R}{j}f(h_1=0,\ldots,h_j=0,h_{j+1}\neq 0,\ldots,h_R\neq 0). \tag{3.17}$$

Separating the term $j=0$

$$f(h_1\neq 0,\ldots,h_R\neq 0)=2^N-\sum_{j=1}^{R}\binom{R}{j}f(h_1=0,\ldots,h_j=0,h_{j+1}\neq 0,\ldots,h_R\neq 0). \tag{3.18}$$

After this, the considerations made so far for *all* the possible configurations can be reproduced to all the sequences for which the first j field components vanish. Note that this gives no information about the other h_i, i.e. some of them may vanish as well and thus we have to put

$$f(h_1=0,\ldots,h_j=0)=\sum_{k=0}^{R-j}\binom{R-j}{k}f(h_1=0,\ldots,h_{j+k}=0,h_{j+k+1}\neq 0,\ldots,h_R\neq 0). \tag{3.19}$$

Once more, the first term in the summatory is separated:

$$f(h_1=0,\ldots,h_j=0,h_{j+1}\neq 0,\ldots,h_R\neq 0)$$
$$=f(h_1=0,\ldots,h_j=0)-\sum_{k=1}^{R-j}\binom{R-j}{k}$$
$$\times f(h_1=0,\ldots,h_{j+k}=0,h_{j+k+1}\neq 0,\ldots,h_R\neq 0). \tag{3.20}$$

Equation (3.18), together with (3.20), constitute a set of interrelated recursive equations, whose solution we have worked out with some labour in appendix A, the result being given by the beautiful expression

$$f(h_1\neq 0,\ldots,h_R\neq 0)=2^N+\sum_{k=1}^{R}(-1)^k\binom{R}{k}f(h_1=0,\ldots,h_k=0) \tag{3.21}$$

and therefore, the accessibilities of the σ patterns are given by

$$A(\sigma^\mu)=\frac{1}{2^N}f(h_1\neq 0,\ldots,h_R\neq 0) \qquad \mu=1,\ldots,N. \tag{3.22}$$

The calculations of this section may be useful in other topics in physics and mathematics due to the fact that binary input patterns may be regarded as the vertices of an *N-dimensional hypercube* or, equivalently, as vectors which go from the centre of the hypercube to its corners. Following this geometrical interpretation, the orthogonality $f(h_1\neq 0,\ldots,h_j\neq 0)$ counts the number of vectors perpendicular to a given set of j mutually orthogonal vectors, $j=1,\ldots,R$, $N=2^R$, and so on. This sort of analysis is applicable, for instance, to the configuration space of Ising models.

3.2. Accessibilities at finite temperature

As we have seen, at zero temperature some of the ξ that do not belong to the set $\{\xi^\mu\}$ can yield $\sigma_j=0$ for one or more j. The chance of having vanishing components makes the number of possible different σ patterns increase from 2^R to 3^R. A way of coping with this is to introduce random noise in the form of finite temperature. Then, the state of the unit σ_j is given by a stochastic function which can take either the value of +1 or −1, with probabilities provided by the sigmoid curve

$$P(\sigma_j=\pm 1)=\frac{1}{1+e^{\mp 2\beta h_j}}. \tag{3.23}$$

In the limit where β goes to infinity, this reproduces a deterministic step function, associated with the 0 and 1 'probabilities'—or rather certainties—when taking the sign function, while for $\beta\to 0$ both probabilities tend to $\frac{1}{2}$, i.e. the system behaves absolutely randomly.

If the process is repeated for all the possible input patterns several times, we can consider average values of each σ unit for every ξ sequence. Let $\langle\sigma\rangle_{\xi=\xi^\mu}$ denote the average of the σ pattern produced by the unary sequence ξ^μ over many repetitions of the whole reading process. Obviously, the lower T, the closer $\langle\sigma\rangle_{\xi=\xi^\mu}$ will be to σ^μ. Therefore, since we are interested in preserving the encoding from ξ^μ to σ^μ (if not always at least on average) the temperature will have to be low.

At $T>0$, owing to the absence of vanishing σ_j, the only possible configurations are the σ^μ, for $\mu=1,\ldots,N$. However, for any fixed μ there are ξ other than the ξ^μ

which end up by giving σ^μ. With respect to the situation at $T=0$, the accessibility of each σ^μ necessarily changes, as patterns which produced one or more zeros will now have to 'decide' among $\{\sigma^\mu, \mu=1,\ldots,N\}$. Since each realization in itself is a merely stochastic result, the only meaningful quantity to give us an idea of these new accessibilities will be the average over many repetitions, that we define as follows:

$$\langle A(\sigma^\mu)\rangle = \frac{\text{cumulative no. of input patterns which have given } \sigma^\mu}{\text{cumulative no. of patterns read } (= \text{no. of repetitions} \times 2^N)}.$$

The result of a simulation (see figure 2) for $N=4$, $R=2$ shows the tendency of all the accessibilities to be equal as the number of repetitions increases, i.e.

$$\langle A(\sigma^\mu)\rangle \to \frac{1}{2^R}.$$

Contrary to other memory retrieval systems, this network has no critical temperature. This means that there is no phase transition in the sense that noise degrades the interactions between processing elements in a continuous way, without leaving any phase where the reproduction of the original process as regards the ξ^μ can be—on average—exact. By (3.23) we obtain

$$\begin{aligned}\langle \sigma_j\rangle_{\xi=\xi^\mu} &= (+1)\times P(\sigma_j^\mu = +1) + (-1)\times P(\sigma_j^\mu = -1)\\ &= \tanh(\beta h_j^\mu) = \tanh\left(\beta\left(\sum_k \omega_{jk}\xi_k^\mu - \theta_j\right)\right).\end{aligned} \tag{3.24}$$

With the components of ξ^μ and the thresholds we are using, this is

$$\begin{aligned}\langle \sigma_j\rangle_{\xi=\xi^\mu} &= \tanh\left(\beta\left(\sum_k \omega_{jk}(2\delta_k^\mu - 1) + \sum_k \omega_{jk}\right)\right)\\ &= \tanh(2\beta\omega_{j\mu}).\end{aligned} \tag{3.25}$$

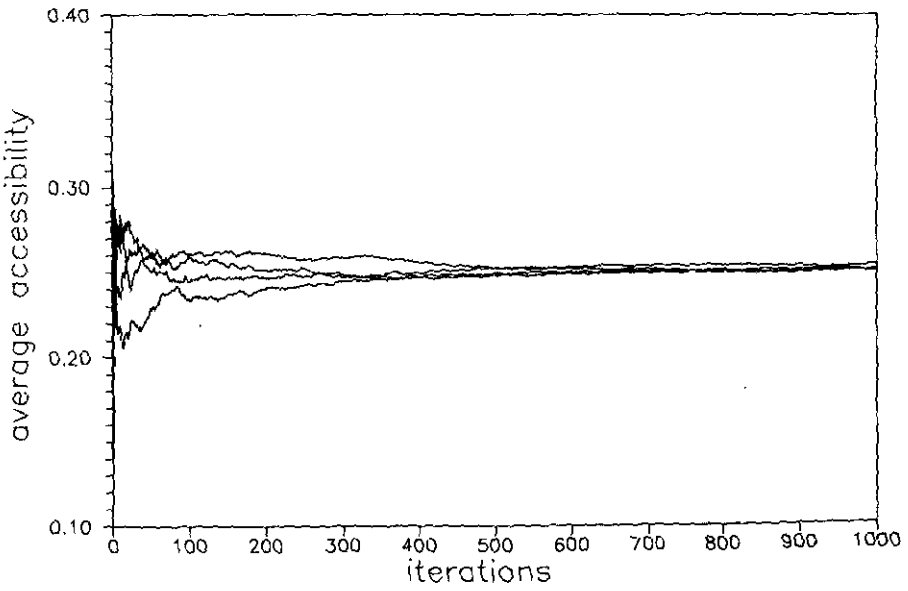

Figure 2. Result of a simulation for $N=4$ at finite $T=0.05$. The curves represent the cumulative average accessibilities of each ξ^μ.

If we look for solutions to $\langle\sigma_j\rangle_{\xi=\xi^\mu}=\sigma_j^\mu=\omega_{j\mu}$, taking into account that for our choice of weights $\omega_{j\mu}$ can be either $+1$ or -1, the equation for β will be in any case

$$1=\tanh 2\beta \tag{3.26}$$

whose only solution is $\beta\to\infty$, i.e. $T=0$. Thus, in this sense, no critical temperature exists. However, this reasoning allows us to find error bounds. The difference between the average obtained and the desired result will be

$$\begin{aligned}\langle\sigma_j\rangle_{\xi=\xi^\mu}-\sigma_j^\mu&=\tanh(2\beta\sigma_j^\mu)-\sigma_j^\mu\\&=\begin{cases}\tanh 2\beta-1 & \text{if } \sigma_j^\mu=+1\\ -\tanh 2\beta+1 & \text{if } \sigma_j^\mu=-1.\end{cases}\end{aligned} \tag{3.27}$$

Hence

$$|\langle\sigma_j\rangle_{\xi=\xi^\mu}-\sigma_j^\mu|=1-\tanh 2\beta. \tag{3.28}$$

If we wish to work in such conditions that

$$|\langle\sigma_j\rangle_{\xi=\xi^\mu}-\sigma_j^\mu|\leqslant\varepsilon \tag{3.29}$$

for a given ε, by the above relations we find that this temperature must have a value satisfying

$$\beta\geqslant\frac{1}{4}\log\frac{2-\varepsilon}{\varepsilon}. \tag{3.30}$$

For example, if, at a given moment, we want our average values to be reliable up to the fourth decimal digit, taking $\varepsilon=10^{-5}$ we get $\beta\geqslant 3.05$ or $T\leqslant 0.33$, which agrees quite well with the behaviour observed in our simulations.

4. Conclusions

Encoding with multilayer neural networks is interpreted as an alternative to supervised learning. This approach makes possible a deeper study of the working of these sorts of networks when an encoding scheme is found. The lack of solution to the minimal encoding problem—when arbitrary input and output alphabets are considered—has led us to the study of other non-optimal set-ups. For several architectures, we have found adequate sets of weights and thresholds giving rise to particularly simple and highly adaptive encoding processes. All our answers take advantage of the full power of multilayer network schemes.

We have also analysed the behaviour of one of our systems when the input supplied does not belong to the initial set, a situation in which the intermediate binary units may no longer yield a definite sign. The procedure followed has consisted in allowing for the occurrence of null values and treating the sequences containing zeros as 'spurious'. Despite the presence of such patterns, we have produced a general method for calculating the accessibility on 'non-spurious' memories when the number of digital units in the input pattern is $N=2^R$, $R\in N$. Further, those unwanted sequences disappear when the level of thermal noise is raised above zero.

As a result, the accessibilities of the 'non-spurious' sequences are modified in such a way that all of them maintain their equiprobability on average. At the same time, as T increases the average values of the outputs become more and more noisy—i.e. away

from the expected result at $T = 0$. Since it turns out that there is no phase transition, this degradation is actually continuous, but if one demands just a certain accuracy in the preservation of the zero-temperature results, an upper bound to the T fulfilling this condition can be found.

Possible applications of our results include all the situations in which multilayer neural networks made of binary units are used, besides processes such as signal encoding-decoding or pattern recognition, which may be understood as particular cases. In addition, some of the methods and equations involving the frequencies of orthogonalities may be useful whenever two-state particle statistical models are considered, and even from a purely mathematical point of view.

Prospects for future developments of these ideas are the pursuit of more amenable expressions for the accessibilities, particularly their dependence on the number of units and on the encoding solution adopted, as well as the design of algorithms aimed at the achievement of learning skills and the improvement of memory access. Moreover, extensions of the encoding to non-binary networks, and to cases in which the number of input patterns is greater than the number of input units, are of maximum interest.

Acknowledgments

This work has been partly supported by Comisión Interministerial de Ciencia y Tecnología (CICYT) contract no. AEN 90-0033. AR is thankful for an FPI fellowship from the Spanish Ministry of Education and Science that has provided financial help during the first stages of this research, and SG acknowledges the awarding of the same type of grant during the last stages.

Appendix A. Accessibilities in terms of orthogonalities

Substituting (3.20) into (3.18) we obtain

$$f(h_1 \neq 0, \ldots, h_R \neq 0)$$
$$= 2^N - \sum_{j=1}^{R} \binom{R}{j} f(h_1 = 0, \ldots, h_j = 0) + (-1)^2 \sum_{j=1}^{R} \sum_{k=1}^{R-j} \binom{R}{j} \binom{R-j}{k}$$
$$\times f(h_1 = 0, \ldots, h_{j+k} = 0, h_{j+k+1} \neq 0, \ldots, h_R \neq 0). \tag{A.1}$$

By recurrent iterations of this sort of substitution in the last term each time, we finally end up with

$$f(h_1 \neq 0, \ldots, h_R \neq 0)$$
$$= 2^N + \sum_{l=1}^{R} (-1)^l \sum_{k_1=1}^{R} \sum_{k_2=1}^{R-k_1} \sum_{k_3=1}^{R-k_1-k_2} \cdots \sum_{k_l=1}^{R-k_1-\ldots-k_{l-1}} \binom{R}{k_1} \binom{R-k_1}{k_2}$$
$$\times \binom{R-k_1-k_2}{k_3} \cdots \binom{R-k_1-\ldots-k_{l-1}}{k_l} f(h_1 = 0, \ldots, h_{k_1+\ldots+k_l} = 0)$$
$$\equiv 2^N + \sum_{l=1}^{R} (-1)^l S_l \tag{A.2}$$

where we have introduced S_l as a shorthand for each l-dependent term in the multiple summatory. Defining the new indices

$$\begin{cases} j_1 \equiv k_1 + \ldots + k_l \\ j_2 \equiv k_1 + \ldots + k_{l-1} \\ j_3 \equiv k_1 + \ldots + k_{l-2} \\ \quad \vdots \\ j_l \equiv k_1 \end{cases} \qquad \begin{cases} l \leq j_1 \leq R \\ l-1 \leq j_2 \leq j_1 - 1 \\ l-2 \leq j_3 \leq j_2 - 1 \\ \quad \vdots \\ 1 \leq j_l \leq j_{l-1} \end{cases}$$

we can put S_l as

$$S_l = \sum_{j_1=l}^{R} \sum_{j_2=l-1}^{j_1-1} \sum_{j_3=l-2}^{j_2-1} \cdots \sum_{j_l=l}^{j_{l-1}-1} \binom{R}{j_l} \binom{R-j_l}{j_{l-1}-j_l} \binom{R-j_{l-1}}{j_{l-2}-j_{l-1}} \cdots \binom{R-j_2}{j_1-j_2} \times f(h_1 = 0, \ldots, h_{j_1} = 0). \tag{A.3}$$

Multiplying and dividing each term by $j_1!\, j_2! \ldots j_{l-1}!$, this becomes

$$S_l = \sum_{j_1=l}^{R} \binom{R}{j_1} f(h_1 = 0, \ldots, h_{j_1} = 0) \sum_{j_2=l-1}^{j_1-1} \binom{j_1}{j_2} \sum_{j_3=l-2}^{j_2-1} \binom{j_2}{j_3} \cdots \sum_{j_l=1}^{j_{l-1}-1} \binom{j_{l-1}}{j_l}. \tag{A.4}$$

Next, successively recalling that

$$\sum_{j=0}^{k} \binom{k}{j} = 2^k$$

and exercising due care with the missing terms in each of the sums occurring we get

$$\begin{aligned} S_l &= \sum_{j_1=l}^{R} \binom{R}{j_1} f(h_1 = 0, \ldots, h_{j_1} = 0) \sum_{j_2=l-1}^{j_1-1} \binom{j_1}{j_2} \cdots \sum_{j_{l-1}=2}^{j_{l-2}-1} \binom{j_{l-2}}{j_{l-1}} (2^{j_{l-1}} - 2) \\ &= \sum_{j_1=l}^{R} \binom{R}{j_1} f(h_1 = 0, \ldots, h_{j_1} = 0) \sum_{j_2=l-1}^{j_1-1} \binom{j_1}{j_2} \cdots \sum_{j_{l-2}=3}^{j_{l-3}-1} \binom{j_{l-3}}{j_{l-2}} \\ &\quad \times (3^{j_{l-2}} - 3 \times 2^{j_{l-2}} + 3) \\ &= \sum_{j_1=l}^{R} \binom{R}{j_1} f(h_1 = 0, \ldots, h_{j_1} = 0) \sum_{j_2=l-1}^{j_1-1} \binom{j_1}{j_2} \cdots \sum_{j_{l-3}=4}^{j_{l-4}-1} \binom{j_{l-4}}{j_{l-3}} \\ &\quad \times (4^{j_{l-3}} - 4 \times 3^{j_{l-3}} + 6 \times 2^{j_{l-3}} - 4) \\ &\quad \vdots \\ &= \sum_{j_1=l}^{R} \binom{R}{j_1} f(h_1 = 0, \ldots, h_{j_1} = 0) \left[(-1)^l \sum_{k=1}^{l} (-1)^k \binom{l}{k} k^{j_1} \right]. \end{aligned} \tag{A.5}$$

Next, let us focus on the quantity in square brackets. Using the notation

$$S(l, j) \equiv \sum_{k=1}^{l} (-1)^k \binom{l}{k} k^j \tag{A.6}$$

one can check the quite remarkable properties

$$S(l, j) = 0 \qquad \text{for } 1 \leq j < l \tag{A.7}$$

$$\sum_{l=1}^{j} S(l, j) = (-1)^j \qquad \text{for } 1 \leq j \tag{A.8}$$

whose proof is given in appendix B. Then, in terms of $S(l,j)$,

$$\sum_{l=1}^{R}(-1)^l S_l = \sum_{l=1}^{R}\sum_{j=1}^{R}\binom{R}{j} f(h_1=0,\ldots,h_j=0)S(l,j) \tag{A.9}$$

where, by the first property, the range of the sum over j has been extended from $j=1$ to R changing nothing. As a result we can write

$$\sum_{l=1}^{R}(-1)^l S_l = \sum_{j=1}^{R}\binom{R}{j} f(h_1=0,\ldots,h_j=0)\sum_{l=1}^{R} S(l,j). \tag{A.10}$$

Moreover, by virtue of the same property the sum over l can be restricted to the range from 1 to j, because the remaining terms give a zero contribution, and then, applying the second one,

$$\sum_{l=1}^{R}(-1)^l S_l = \sum_{j=1}^{R}\binom{R}{j} f(h_1=0,\ldots,h_j=0)(-1)^j. \tag{A.11}$$

Consequently,

$$f(h_1\neq 0,\ldots,h_R\neq 0) = 2^N + \sum_{j=1}^{R}(-1)^j\binom{R}{j} f(h_1=0,\ldots,h_j=0) \tag{A.12}$$

which is (3.21).

Appendix B. Proof of two properties

B.1. Proof of $S(l,j)=0$, $1\leqslant j<l$

We start by considering the function

$$y_{(l,0)} \equiv (1-x)^l = \sum_{k=1}^{l}(-1)^k\binom{l}{k}x^k. \tag{B.1}$$

Then, we make the definitions

$$y_{(l,1)} \equiv \frac{\mathrm{d}}{\mathrm{d}x}y_{(l,0)}(x) = \sum_{k=1}^{l}(-1)^k\binom{l}{k}kx^{k-1} \tag{B.2}$$

$$y_{(l,j+1)} \equiv \frac{\mathrm{d}}{\mathrm{d}x}(xy_{(l,j)}(x)) \qquad j\geqslant 1 \tag{B.3}$$

the second one being a recurrent constructive rule. In terms of these functions, one has

$$S(l,0) = y_{(l,0)}(1) - 1 \qquad l\geqslant 1 \tag{B.4}$$

$$S(l,j) = y_{(l,j)}(1) \qquad j\geqslant 1 \qquad l\geqslant 1. \tag{B.5}$$

Since $y_{(l,0)}=0$, one realizes that

$$S(l,0) = -1 \qquad l\geqslant 1. \tag{B.6}$$

The next step is to show that $S(l,j)=0$ for $1\leqslant j<l$. By taking successive derivatives, it is not difficult to see that $y_{(l,k)}(x)$ is a sum of terms proportional to $(1-x)^{l-k}$, with $1\leqslant k<l$. Therefore

$$y_{(l,j)}(1) = 0 = S(l,j) \qquad \text{for } 1\leqslant j<l. \tag{B.7}$$

B.2. Proof of $\sum_{l=1}^{j} S(l,j) = (-1)^j$, $j \geqslant 1$

$$\sum_{l=1}^{j} S(l,j) = \sum_{l=1}^{j} \sum_{k=1}^{l} (-1)^k \binom{l}{k} k^j. \tag{B.8}$$

Given that the binomial coefficients vanish for $l < k$, the second sum can be extended to the range from $k = 1$ to j and interchanged with the first afterwards

$$\sum_{l=1}^{j} S(l,j) = \sum_{k=1}^{j} (-1)^k k^j \sum_{l=1}^{j} \binom{l}{k}. \tag{B.9}$$

Further, by the same reasoning the l-summatory may now be restricted to $k \leqslant l$. Then we obtain

$$\sum_{l=k}^{j} \binom{l}{k} = \binom{j+1}{k+1}. \tag{B.10}$$

Replacing this in the previous expression and making the index renaming

$$\begin{cases} N \equiv j+1 \\ r \equiv k+1 \end{cases} \tag{B.11}$$

we arrive at

$$\begin{aligned} \sum_{l=1}^{j} S(l,j) &= \sum_{r=2}^{N} (-1)^{r-1} (r-1)^{N-1} \binom{N}{r} \\ &= -\sum_{r=0}^{N} (-1)^r (r-1)^{N-1} \binom{N}{r} + (-1)^{N-1} \binom{N}{0}. \end{aligned} \tag{B.12}$$

The first term vanishes by a known property (Gradshteyn and Ryzhik 1980, formula [0.154(6)]) and what remains reads

$$\sum_{l=1}^{j} S(l,j) = (-1)^j. \tag{B.13}$$

References

Amit D J, Evans M R, Horner H and Wong K Y M 1990 *J. Phys. A: Math. Gen.* **23** 3361
Bacci S, Mato G and Parga N 1990 *J. Phys. A: Math. Gen.* **23** 1801
Denby B 1990 Fermilab *Preprint* Conf-90/94
Denby B, Campbell M, Bedeschi F, Chris N, Bowers C and Nesti F 1990 Fermilab *Preprint* Conf-90/20
Derrida B, Gardner E and Zippelius A 1987 *Europhys. Lett.* **4** 167
Gradshteyn I S and Ryzhik I M 1980 *Table of Integrals, Series and Products* (New York: Academic)
Herz A V M, Li Z and van Hemmen J L 1991 *Phys. Rev. Lett.* **66** 1370
Hertz J and Palmer R G 1988 Duke University lecture notes
Hopfield J J 1982 *Proc. Natl Acad. Sci. USA* **79** 2554
—— 1984 *Proc. Natl Acad. Sci. USA* **81** 3088
Hopfield J J, Feinstein D I and Palmer R G 1983 *Nature* **304** 158
Hopfield J J and Tank D W 1985 *Biol. Cybernetics* **52** 141
Little W A 1974 *Math. Biosc.* **19** 101
Minsky M and Papert S 1969 *Perceptrons* (Cambridge, MA: MIT Press)
Nakamura T and Nishimori H 1990 *J. Phys. A: Math. Gen.* **23** 4627

Parga N and Virasoro M A 1986 *J. Physique* **47** 1857
Rumelhart D E, Hinton G E and Williams R J 1986 *Nature* **323** 533
Rumelhart D E, McClelland J L and the PDP research group 1986 *Parallel Distributed Processing* (Cambridge, MA: MIT Press) ch 8
Schempp W 1984 *Proc. of the A.M.S.* **92** 345
—— 1989 *Results in Math.* **16** 103
Stimpfl-Abele G and Garrido L 1991 *Comput. Phys. Commun.* **64** 46