Capacity of the single-layer perceptron and minimal trajectory training algorithms

1993 J. Phys. A: Math. Gen. 26 3757

(http://iopscience.iop.org/0305-4470/26/15/025)

J. Phys. A: Math. Gen. 26 (1993) 3757–3773. Printed in the UK

Capacity of the single-layer perceptron and minimal trajectory training algorithms

D Saad†
Faculty of Engineering, Tel Aviv University, 69978, Israel

Received 13 April 1992, in final form 10 May 1993

Abstract. The entire set of binary vectors to be stored using a single-layer perceptron can be divided into two groups, one for which the output neuron state consistently equals one of the input neuron states and a second for which the output neuron state consistently negates the same input neuron. The capacity of the single-layer perceptron depends on the ratio between these two groups. This dependence is examined via statistical mechanical methods, producing the probability of obtaining a linearly separable solution for a random selection of input–output relations, for a given value of the above ratio. This probability is extremely useful for designing recurrent neural network training algorithms. These algorithms make use of the obtained results to select the most probable internal representations to be realized in such nets. Moreover, the distribution of the linearly separable binary functions enables us to obtain a good estimate for the total number of linearly separable binary functions for a certain number of input neurons, a task considered as a hard computational problem. Additional incentives for carrying out the calculation are understanding the capacity of simple nets for certain types of input–output correlations and laying the foundations for analysing some constructive training algorithms such as the tiling and upstart algorithms. All results show consistency with existing theoretical results.

1. Introduction

Examining the single-layer perceptron with N neuron input layers and a single output neuron, one easily notices that there are two cases for which the desired input–output relations can always be realized:

(i) the output neuron state is consistently identical to a certain (fixed) input neuron state for each and every input vector;

(ii) the output neuron state is consistently opposite to a certain (fixed) input neuron state for each and every input vector.

Obviously these two cases can be achieved by defining the weight vector connecting the output neuron to the above mentioned input neuron to be greater in its absolute value than the summation of the absolute values of all other neurons, while its sign selects between the two cases‡. Each of these cases also remains linearly separable if we randomly select one of the input vectors to produce an output result which opposes the said resemblance rule (an irregular vector). This can be easily obtained by defining the incoming weights with respect to the output neuron according to the irregular vector (besides the weight element connecting the output neuron to the input neuron which determines the similarity) as well

† Current address: Department of Physics, University of Edinburgh, King's Buildings, Mayfield Road, Edinburgh EH9 3JZ, UK.

‡ A special case would be a constant +1 or −1 output, obtained by selecting the desired output to resemble the constant dummy neuron representing the threshold via its weight.

0305-4470/93/153757+17$07.50

as defining the threshold in accordance with the desired result. For example, if the irregular vector is v^q and the desired output is

$$v^{p,\text{out}} = \begin{cases} -v_1^p & \text{if } p = q \\ v_1^p & \text{otherwise} \end{cases}$$

the weight vector for separating the two cases is then of the form

$$W_j = \begin{cases} N - \epsilon & \text{for } j = 1 \\ -v_1^q v_j^q & \text{otherwise, including } j = 0 \end{cases}$$

where $\boldsymbol{W}$ represents the weight vector, ϵ is a small positive constant obeying $0 < \epsilon < 2$ and the index 0 is related to the dummy neuron used for the output neuron's threshold.

Intuitively, when we require more input vectors to be irregular in relation to the defined rule we increase the burden on the weight vector, since it has to carry more and more information for selecting the irregulars from the regular group of input–output relations. Thus, we should expect that by increasing the number of irregulars we decrease the probability of implementing the said relation via a single-layer perceptron. This probability is expected to decrease until the number of irregular members surpasses the number of regular ones, reversing the regular and irregular groups (now with respect to the input–output relations negating the one with which we originally started).

In this paper we examine the dependence of the single-layer perceptron capacity on the ratio between the regular and irregular input–output relations using Gardner's method [1], giving us insight into understanding the storage capabilities of correlated patterns of a certain type. This dependence enables us to compute the probability of selecting a linearly separable combination of 'regular' and 'irregular' input–output relations for a given ratio between the two groups. In addition, by summing the number of linearly separable binary functions over all possible ratios one obtains the total number of linearly separable functions, for a given number N of input neurons, a task considered as a hard computational problem [18].

A comparison between the computed and theoretical results for both the probabilities of realizing a certain input–output relation, with a given correlation ratio, and the total number of linearly separable groups shows good correlation with other published results related to specific values of N.

An additional incentive for computing these probabilities is that they form the basic tools for analysing constructive algorithms such as tiling [2] and upstart [3]. These algorithms are based on storing the input–output relations of the various training patterns in parts, enhancing the input–output correlation for successive layers.

The last section of this paper explains the relation between the new results and the minimal trajectory algorithm [4]. The basic idea is to use the new results to construct a recurrent neural network training algorithm based on prior selection of the most probable internal representations to be realized in this architecture. The probabilities for the internal representations selection are then directly related to results obtained in this work.

2. Computing the capacity

For computing the capacity we use Gardner's method [1], based on measuring the normalized weight space volume enabling the storage of p vectors:

$$V = \frac{\int \mathrm{d}w \left[\prod_{\mu=1}^{p} \Theta(\zeta^\mu (1/\sqrt{N}) \sum_{j=1}^{N} w_j \xi_j^\mu - \kappa)\right] \delta\left(\sum_{j=1}^{N} w_j^2 - N\right)}{\int \mathrm{d}w\, \delta\left(\sum_{j=1}^{N} w_j^2 - N\right)} \tag{1}$$

where μ is the pattern index, j is the site index, ζ^μ and ξ^μ are the output neuron state and the input vector state respectively, related to pattern index μ, w is the set of weights, N the number of input neurons, κ is the margin size and Θ a step function. The denominator as well as the last term in the numerator result from an additional restriction for the set of weights $\sum_{j=1}^{N} w_j^2 = N$. The numerator in equation (1) represents the volume of the weight space enabling the storage of p patterns while the denominator represents the volume of the entire relevant weight space.

Since the statistically relevant quantity is the average over the pattern distribution, i.e. of the logarithm of V, we follow Gardner by introducing replicas for the weights, averaging the value of V^n

$$\langle\langle V^n \rangle\rangle = \left\langle\!\!\left\langle \frac{\prod_{\alpha=1}^{n} \int \mathrm{d}w^\alpha \left[\prod_{\mu^+=1}^{p^+} \Theta^\alpha(\zeta^{\mu^+}, \xi^{\mu^+})\right]\left[\prod_{\mu^-=1}^{p^-} \Theta^\alpha(\zeta^{\mu^-}, \xi^{\mu^-})\right] \delta\left(\sum_{j=1}^{N} (w_j^\alpha)^2 - N\right)}{\prod_{\alpha=1}^{n} \int \mathrm{d}w^\alpha \, \delta\left(\sum_{j=1}^{N} (w_j^\alpha)^2 - N\right)} \right\rangle\!\!\right\rangle \tag{2}$$

where α is the replica index and

$$\Theta^\alpha(\zeta^{\mu^\pm}, \xi^{\mu^\pm}) = \Theta\left(\zeta^{\mu^\pm} \frac{1}{\sqrt{N}} \sum_{j=1}^{N} w_j^\alpha \xi_j^{\mu^\pm} - \kappa\right). \tag{3}$$

Since for all patterns indexed μ^+ the output neuron state equals a certain input neuron (which we shall define as $\xi_1^{\mu^+}$ with no loss of generality), while for all patterns indexed μ^- the output neuron state equals $-\xi_1^{\mu^-}$, we can express the Θ functions explicitly as

$$\Theta\left(\zeta^{\mu^\pm} \frac{1}{\sqrt{N}} \sum_{j=1}^{N} w_j^\alpha \xi_j^{\mu^\pm} - \kappa\right) = \Theta\left(\pm \xi_1^{\mu^\pm} \frac{1}{\sqrt{N}} \sum_{\forall j, j \neq 1}^{N} w_j^\alpha \xi_j^{\mu^\pm} - \kappa^\pm\right) \tag{4}$$

where $\kappa^\pm \equiv \kappa \mp w_1^\alpha / \sqrt{N}$.

Replacing the Θ function by its integral form

$$\Theta(z_\alpha^{\mu^\pm} - \kappa^\pm) = \int_{\kappa^\pm}^{\infty} \frac{\mathrm{d}\lambda_\alpha^{\mu^\pm}}{2\pi} \int \mathrm{d}x_\alpha^{\mu^\pm} \, \mathrm{e}^{\mathrm{i}x_\alpha^{\mu^\pm} \lambda_\alpha^{\mu^\pm}} \mathrm{e}^{-\mathrm{i}x_\alpha^{\mu^\pm} z_\alpha^{\mu^\pm}} \tag{5}$$

where

$$z_\alpha^{\mu^\pm} \equiv \pm \xi_1^{\mu^\pm} \frac{1}{\sqrt{N}} \sum_{\forall j, j \neq 1}^{N} w_j^\alpha \xi_j^{\mu^\pm} \tag{6}$$

we can average the expression over all patterns and replicas in the limit of large N to obtain

$$\lim_{N \to \infty} \left\langle\!\!\left\langle \prod_{\alpha, \mu^\pm} \mathrm{e}^{-\mathrm{i}x_\alpha^{\mu^\pm} z_\alpha^{\mu^\pm}} \right\rangle\!\!\right\rangle$$

$$= \prod_{\mu^\pm} \exp\left(-\frac{1}{2}\left[1 - \frac{(w_1^\alpha)^2}{N}\right] \sum_\alpha (x_\alpha^{\mu^\pm})^2 - \sum_{\alpha < \beta} q_{\alpha\beta} x_\alpha^{\mu^\pm} x_\beta^{\mu^\pm}\right) \tag{7}$$

where $q_{\alpha\beta} \equiv (1/N) \sum_{j \neq 1} w_j^\alpha w_j^\beta$ is a new order parameter (note that $q_{\alpha\alpha} \equiv 1 - (w_1^\alpha)^2/N$).

The next step is to rewrite the average over the product of Θ functions and to replace the restriction for the value of $\sum_j (w_j^\alpha)^2$ and for the definition of $q_{\alpha\beta}$ by the following δ functions:

$$\delta\left(\sum_{j=1}^{N}(w_j^\alpha)^2 - N\right) = \int \frac{\mathrm{d}E_\alpha}{4\pi \mathrm{i}} \exp\left[\left(N - (w_1^\alpha)^2\right)\tfrac{1}{2}E_\alpha - \tfrac{1}{2}E_\alpha \sum_{j\neq 1}(w_j^\alpha)^2\right] \tag{8}$$

and

$$\delta\left(q_{\alpha\beta} - \frac{1}{N}\sum_{j\neq 1} w_j^\alpha w_j^\beta\right) = N \int \frac{\mathrm{d}F_{\alpha\beta}}{2\pi \mathrm{i}} \exp\left[-NF_{\alpha\beta}q_{\alpha\beta} + F_{\alpha\beta}\sum_{j\neq 1} w_j^\alpha w_j^\beta\right] \tag{9}$$

where E_α and $F_{\alpha\beta}$ are the integration variables related to certain replicas α and β. One can then replace equation (2) by the following equation

$$\langle\langle V^N\rangle\rangle = \frac{\int(\prod_\alpha \mathrm{d}E_\alpha)(\prod_{\alpha<\beta}\mathrm{d}q_{\alpha\beta}\,\mathrm{d}F_{\alpha\beta})\mathrm{e}^{(N-1)G}}{\int(\prod_\alpha \mathrm{d}E_\alpha)\mathrm{e}^{(N-1)H}} \tag{10}$$

where G and H are defined as

$$\begin{aligned} G \equiv{}& \frac{p^+}{N-1}\log\left[\int_{\kappa^+}^{\infty}\left(\prod_\alpha \frac{\mathrm{d}\lambda_\alpha^+}{2\pi}\right)\int\left(\prod_\alpha \mathrm{d}x_\alpha^+\right)\mathrm{e}^{K^+}\right] \\ &+ \frac{p^-}{N-1}\log\left[\int_{\kappa^-}^{\infty}\left(\prod_\alpha \frac{\mathrm{d}\lambda_\alpha^-}{2\pi}\right)\int\left(\prod_\alpha \mathrm{d}x_\alpha^-\right)\mathrm{e}^{K^-}\right] \\ &+ \log\left\{\int\left(\prod_\alpha \mathrm{d}w^\alpha\right)\exp\left[-\sum_\alpha \tfrac{1}{2}E_\alpha(w^\alpha)^2 - \sum_{\alpha<\beta}F_{\alpha\beta}w^\alpha w^\beta\right]\right\}^{N-1} \\ &+ \frac{1}{2}\sum_\alpha E_\alpha\frac{N-(w_1^\alpha)^2}{N-1} + \sum_{\alpha<\beta}F_{\alpha\beta}q_{\alpha\beta}\frac{N}{N-1} \end{aligned} \tag{11}$$

$$H \equiv \log\left\{\int\left(\prod_\alpha \mathrm{d}w^\alpha\right)\exp\left[-\sum_\alpha \frac{E_\alpha}{2}(w^\alpha)^2 + \frac{1}{2}\sum_\alpha E_\alpha(N-(w_1^\alpha)^2)/(N-1)\right]\right\} \tag{12}$$

and

$$K^\pm \equiv \mathrm{i}\sum_\alpha x_\alpha^\pm \lambda_\alpha^\pm - \frac{1}{2}\left[1 - \frac{(w_1^\alpha)^2}{N}\right]\sum_\alpha (x_\alpha^\pm)2 - \sum_{\alpha<\beta} q_{\alpha\beta}x_\alpha^\pm x_\beta^\pm \tag{13}$$

where $p^\pm$ represents the number of patterns to be stored related to the two different groups, the group of vectors in which the desired output resembles the first input neuron state (indexed $+$) and the group in which the output state negates the said neuron state (indexed $-$). The j index of the various weights has been eliminated since it is a dummy index common to all of the elements.

Applying the replica-symmetric ansatz:

$$q_{\alpha\beta} = q \qquad F_{\alpha\beta} = F \qquad E_\alpha = E \tag{14}$$

we can replace K by the following expression:

$$K^{\pm} = \mathrm{i}\sum_{\alpha} x_{\alpha}^{\pm}\lambda_{\alpha}^{\pm} - \frac{1}{2}\left[1 - \frac{(w_1^{\alpha})^2}{N} - q\right]\sum_{\alpha}(x_{\alpha}^{\pm})^2 - \frac{q}{2}\sum_{\alpha}(x_{\alpha}^{\pm})^2. \tag{15}$$

Since the saddle point defines the optimal perceptron (in the limit of $q \to 1 - w_1^2/N$) we calculate the saddle-point equations:

$$\frac{\partial G}{\partial E} = 0 \qquad \frac{\partial G}{\partial F} = 0 \qquad \frac{\partial G}{\partial q} = 0 \tag{16}$$

defining $\omega \equiv w_1/\sqrt{N}$ as the normalized value of the weight w_1 and β^+ and β^- in the following manner

$$\beta^{\pm} \equiv p^{\pm}/\alpha_c(N-1) \tag{17}$$

$$\beta^+ + \beta^- = 1. \tag{18}$$

Carrying out the calculation for the $q \to 1 - \omega^2$ limit which represents the maximal capacity limit one obtains the following expression for α_c†

$$\alpha_c = \left\{\beta^+ \int_{-(\kappa-\omega)/\sqrt{1-\omega^2}}^{\infty} \frac{\mathrm{d}t}{\sqrt{2\pi}} \mathrm{e}^{-t^2/2}\left[t + \frac{\kappa-\omega}{\sqrt{1-\omega^2}}\right]^2 + \beta^- \int_{-(\kappa+\omega)/\sqrt{1-\omega^2}}^{\infty} \frac{\mathrm{d}t}{\sqrt{2\pi}} \mathrm{e}^{-t^2/2}\left[t + \frac{\kappa+\omega}{\sqrt{1-\omega^2}}\right]^2\right\}^{-1}. \tag{19}$$

Clearly, for $\omega = 0$ we retrieve Gardner's capacity expression

$$\alpha_c = \left\{\int_{-\kappa}^{\infty} \frac{\mathrm{d}t}{\sqrt{2\pi}} \mathrm{e}^{-t^2/2}[t+\kappa]^2\right\}^{-1}. \tag{20}$$

Equation (19) still includes a dependence on the value of the normalized weight ω. Since we want the maximal capacity, i.e. the value of α_c for the optimal ω, we should maximize α_c with respect to ω. Applying $\partial\alpha_c/\partial\omega = 0$ and using the identity $\beta^+ + \beta^- = 1$ one obtains the following expression for $\beta^+(\omega)$ related to the maximal capacity α_c

$$\beta^+ = \left\{\int_{-(\kappa+\omega)/\sqrt{1-\omega^2}}^{\infty} \mathrm{d}t\, \mathrm{e}^{-t^2/2}\left[t + \frac{\kappa+\omega}{\sqrt{1-\omega^2}}\right]\left[\frac{\omega\kappa+1}{(1-\omega^2)^{3/2}}\right]\right\} \times \left\{\int_{-(\kappa+\omega)/\sqrt{1-\omega^2}}^{\infty} \mathrm{d}t\, \mathrm{e}^{-t^2/2}\left[t + \frac{\kappa+\omega}{\sqrt{1-\omega^2}}\right]\left[\frac{\omega\kappa+1}{(1-\omega^2)^{3/2}}\right] - \int_{-(\kappa-\omega)/\sqrt{1-\omega^2}}^{\infty} \mathrm{d}t\, \mathrm{e}^{-t^2/2}\left[t + \frac{\kappa-\omega}{\sqrt{1-\omega^2}}\right]\left[\frac{\omega\kappa-1}{(1-\omega^2)^{3/2}}\right]\right\}^{-1}. \tag{21}$$

Using equations (19) and (20) one can derive the optimal β and α_c expressions for each and every selection of the normalized weight ω. One should note that the same result

† This explicit expression is a special case of the general capacity expression presented by Nadal [4].

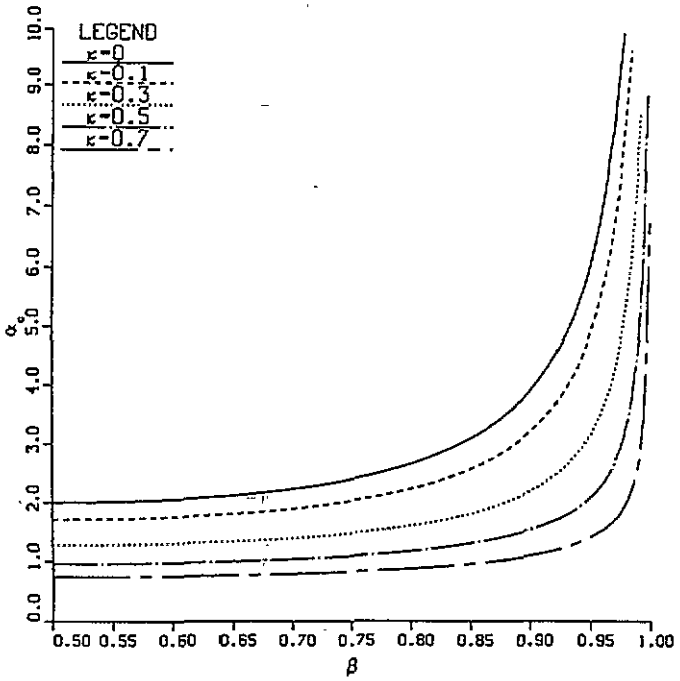

Figure 1. The capacity α_c with respect to the ratio β for several values of κ.

can be obtained by referring to ω as one of the free parameters of the expression for G, demanding an extra saddle-point requirement $\partial G/\partial\omega = 0$.

The dependence of α_c on the ratio β^+ (which we shall define from now on as β since all values are symmetric with respect to β^+ and β^-) is shown in figure 1. Note the similarity between this graph and the dependence of α_c on the magnetization parameter m (representing the probability for a $+1$ bit in the set of stored patterns) presented in [1]. This similarity results from the fact that both of them represent the amount of information stored by the system.

It is important to note that using equation (19) one can compute the value of α_c with respect to *any given ratio β and enforced neuron weight ω* (i.e. not only the optimal weight corresponding to a certain ratio defined by equation (21)), a fact that will be very helpful in the forthcoming section. The optimal values for the weight ω and for q with respect to the ratio β are presented in figure 2. The capacity α_c with respect to various values of the ratio β computed for several enforced values of ω is shown in figure 3. A three-dimensional graph presenting the value of α_c with respect to different values of β and ω is presented in figure 4.

3. Computing the probability of obtaining a linearly separable binary function selecting an arbitrary binary function with a certain ratio β

A binary function is defined by an exhaustive set of 2^N binary vectors (of length N) and the binary output neuron representations related to them. Each and every function is characterized by a certain ratio β, representing the fraction of the entire set of vectors for which the output neuron state equals (or negates, according to our definition) the state of

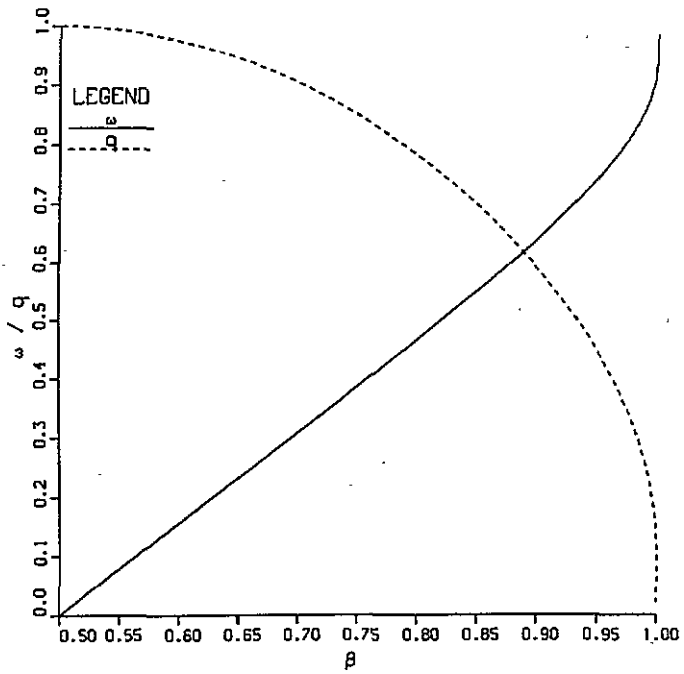

Figure 2. The dependence of ω and q with respect to the ratio β.

a certain input neuron. In this section we will use the capacity expressions obtained in the previous section to derive the probability of obtaining a linearly separable function for a given value of β. This will enable us to obtain a good approximation for the number of linearly separable binary functions related to a certain number N of input neurons†. The calculation of the probability of linear separability is performed by counting the number of linearly separable functions that one can store in a single-layer perceptron based on the capacity computation of the previous section.

One of the main incentives for carrying out the probability computation is its importance for speeding up neural network training procedures by limiting our search for the proper system representations‡. The proper system representations are expected to produce the right input–output relations as well as to be realizable. We therefore search representations related to a feasible solution in those regions of representation space which show a high probability of producing a linearly separable function.

Starting with the exhaustive ensemble of $B \equiv 2^N$ vectors we will first compute the number of linearly separable functions for a given β. Since one can always store $A(\beta') \equiv \alpha_c(\beta')(N-1)$ vectors§ with a certain value β' (defined by equation (19)), we

† An exact solution for this problem can be achieved using optimization techniques such as the simplex method, however, it requires large computational resources and grows exponentially with the number of neurons, already becoming unfeasible for a relatively low number of neurons.

‡ A system representation is the vector representation of a system of neurons with a certain configuration and a certain weight matrix obtained by presenting to the system an input training vector. For feed-forward networks the system representations include representations of the various layers while for recurrent nets they include the vector representations related to the various time steps.

§ Obviously this number is always limited by the maximal number of such vectors that actually exist in the training set.

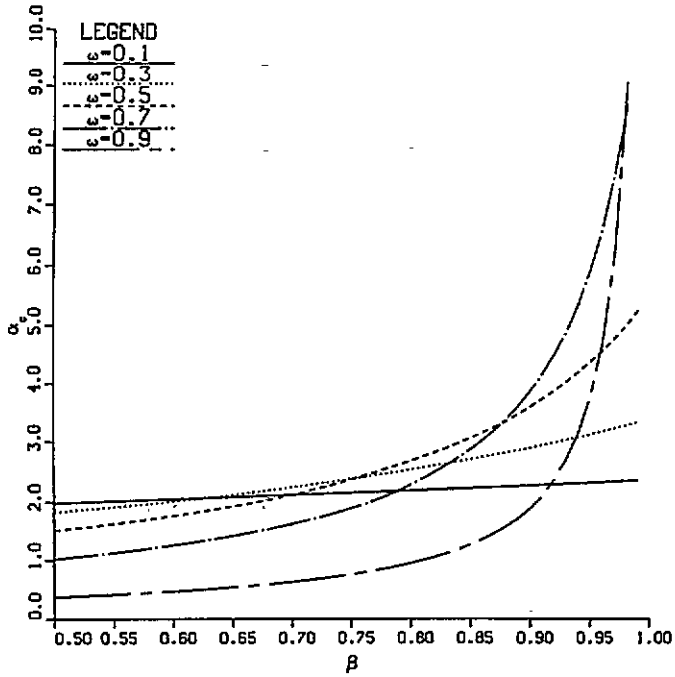

Figure 3. The capacity α_c with respect to the ratio parameter β and several enforced values of the weight ω.

will have a total of $C_{A(\beta')}^{\beta' A(\beta')}$ possible selections of various functions with ratio β', formed out of the selected A vectors, by assigning the different vectors to the regular and irregular groups, $\beta' A$ and $(1-\beta')A$ respectively (the output states for the remaining $B-A$ vectors are now defined by the unique set of weights required for the storage of the first A vectors). However, the selection of A vectors with a value β' does not determine the *actual* ratio of the entire ensemble B since it determines the ratio β'' between regular and irregular groups *only for the first A selected vectors.* We do know, however, that a linearly separable function was selected *for all of the vectors in the ensemble* since an actual set of weights was selected, and that the value of the weight ω related to one of the input neurons was determined by the optimal weight $\omega(\beta')$ required to achieve the optimal capacity for the value β' (equation (21)).

In order to compute the number of functions obtained with an *overall value* β, we define a probability function $\Phi(\beta, \beta', B)$ representing the probability of obtaining an *overall value* β when choosing and storing any A vectors out of the exhaustive ensemble of B vectors with a ratio β' between the two groups of vectors in A. To simplify the notation we will redefine the function Φ with respect to two new arguments $\Phi^{\omega}(\beta', Z)$. The argument $Z \equiv B - A(\beta')$ is the number of remaining vectors after *selecting* A vectors to be stored in the network. The second argument is defined by $\beta' \equiv (\beta B - \beta' A(\beta'))/(B - A(\beta'))$; it is the value of the ratio for the remaining $B-A$ vectors which gives the ratio β for the entire set B. The weight ω is enforced between the output neuron and the correlated input neuron by the previous selection of A vectors. We can therefore compute the number of binary functions for the entire set B with a certain ratio β to be formed by selecting a certain set of $A(\beta')$ vectors as

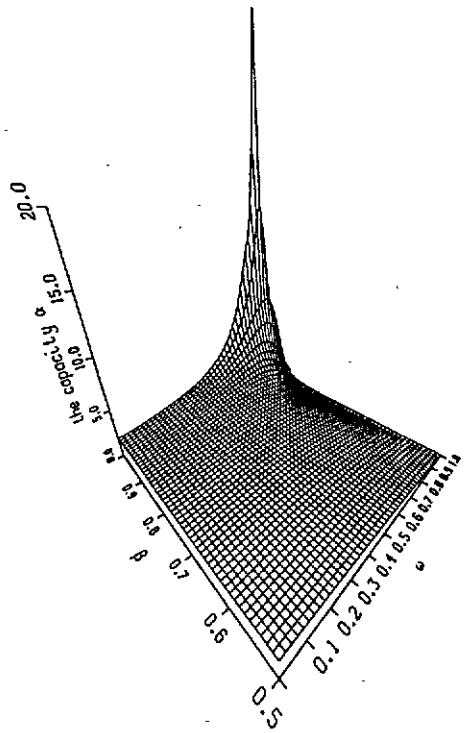

Figure 4. A three-dimensional graph presenting the value of α_c with respect to different values of the ratio parameter β and the weight ω.

$$C_{A(\beta')}^{\beta' A(\beta')} \Phi^{\omega}(\beta'', Z). \tag{22}$$

However, equation (22) computes the number of functions to be formed out of *a certain selection* and storage of A vectors while one can select a different set of A vectors to be stored and obtain different functions. By selecting a new set of vectors one should prevent double counting of functions generated by the various selections†. Thus, in order to obtain a lower bound for the number of functions one should enforce *at least* for one of the vectors to represent an input–output relation *opposite* to the one represented by the same vector in previous selections (thus creating necessarily a different function). This restriction reduces the number of functions obtained by the second selection to $C_{A(\beta')}^{\beta' A(\beta')-1} \Phi^{\omega}(\beta'', Z)$, by the third selection to $C_{A(\beta')}^{\beta' A(\beta')-2} \Phi^{\omega}(\beta'', Z)$ etc. Since any selection of an optimal weight $\omega(\beta')$ enables us to select functions with any ratio $\hat{\beta} > \beta'$ as well‡ we can include in our summations all functions generated by selection of vectors with ratio $\hat{\beta} > \beta'$. All these functions differ from one another by the modification of the ratio $\hat{\beta}$ (for a given weight $\omega(\beta')$) and by different weights $\omega(\beta')$ (for a given ratio $\hat{\beta}$). Accumulating all of the functions of value β generated by various selections of $A(\beta')$ vectors therefore results in the following expression:

† Selecting A input vectors and their related output neuron representations fixes the rest of the input–output relations. A different selection of the A vectors to be stored might result in an overall similar function if the resulting weight matrix assigns the 'free' vectors to generate input–output relations, similar to those defined in previous selections, in the various cases.

‡ As suboptimal capacity for $\beta' > \frac{1}{2}$. The same arguments can be used for the case of $\beta' < \frac{1}{2}$ by replacing the decrement of the number of vectors by an increment and the inequality by $\hat{\beta} < \beta'$.

$$\sum_{n=0}^{A(\beta')} C^n_{A(\beta')} \Phi^{\omega}(\beta'', Z) = 2^{A(\beta')} \Phi^{\omega}(\beta'', Z). \tag{23}$$

Since the number of vectors to be stored A is limited by the number of such vectors that actually exist in the training set we might obtain an unfeasible number of functions. Since the total number of possible functions is $C_B^{\beta' B}$ for the ratio β', equation (23) becomes

$$\text{Min}\{C_B^{\beta' B}, 2^{A(\beta')}\} \Phi^{\omega}(\beta'', Z). \tag{24}$$

The probability $P(\beta, B)$ of obtaining a linearly separable function selecting a random function for the entire set of B vectors with ratio β can now be calculated. It is, in fact, the fraction of linearly separable functions for all possible selections of subsets of vectors with a ratio β' to be stored by the perceptron resulting with an overall β ratio, with respect to the total number of possible functions with a similar ratio β:

$$P(\beta, B) = \frac{\text{Min}\{C_B^{\beta B}, \sum_{\beta'=0}^{1} \text{Min}\{C_B^{\beta' B}, 2^{A(\beta')}\} \Phi^{\omega}(\beta'', Z)\}}{C_B^{\beta B}}. \tag{25}$$

We have also used here the fact that $C_B^{\beta B}$ is the maximum number of possible functions for the ratio β.

A summation over *all* possible β values gives $LS(N)$ the number of *all* possible linearly separable functions for N input neurons.

$$LS(N) = \sum_{\beta} \text{Min}\left\{ C_B^{\beta B}, \sum_{\beta'=0}^{1} \text{Min}\{C_B^{\beta' B}, 2^{A(\beta')}\} \Phi^{\omega}(\beta'', Z) \right\}. \tag{26}$$

Both of the expressions (25) and (26), which are, in fact, the target of our calculation, involve the two functions $A(\beta')$ and $\Phi^{\omega}(\beta'', Z)$. The capacity $A(\beta')$ can be easily computed using equation (19), however, estimating the probability $\Phi^{\omega}(\beta'', Z)$ requires additional computation, as follows.

A certain selection of ratio β' and a certain choice of $A(\beta')$ vectors to be stored determines a certain linearly separable function and a certain weight ω (required to achieve this capacity). One can express the probability $\Phi^{\omega}(\beta'', Z)$ as the ratio between the number of linearly separable functions with a ratio β'' to be formed from an ensemble of Z vectors in a perceptron *with a given weight* ω, and the *total number* of linearly separable functions with all possible ratios that can be formed in the same conditions (Z vectors and one given weight ω). Applying similar considerations to those utilized to obtain equation (26), examining this time an ensemble of Z vectors with a *previously defined weight* $\omega(\beta')$, one obtains the following expression for the probability $\Phi^{\omega}(\beta'', Z)$:

$$\Phi^{\omega}(\beta'', Z) = \frac{\text{Min}\{C_Z^{\beta'' Z}, \sum_{\gamma=0}^{1} \text{Min}\{C_Z^{\gamma Z}, 2^{D(\gamma,\omega)}\} \Phi^{\omega}(\gamma'', Z - D(\gamma, \omega))\}}{\sum_{\gamma'=0}^{1} \text{Min}\{C_Z^{\gamma' Z}, \sum_{\gamma=0}^{1} \text{Min}\{C_Z^{\gamma Z}, 2^{D(\gamma,\omega)}\} \Phi^{\omega}(\hat{\gamma''}, Z - D(\gamma, \omega))\}} \tag{27}$$

where $D(\gamma, \omega) \equiv \alpha_c(\gamma, \omega(\beta'))(N-1)$ is the capacity of a perceptron for a certain ratio γ and a *given weight* ω (defined by equation (21)†. Once D vectors with ratio γ have been chosen (out of the ensemble Z) we require a ratio

$$\gamma'' \equiv \frac{\beta'' Z - \gamma D(\gamma, \omega)}{Z - D(\gamma, \omega)}$$

† Note that here we make use of the general form of the capacity expression obtained previously.

for the rest of the $(Z-D)$ vectors in order to obtain an overall β'' ratio (in a similar manner we require a ratio

$$\hat{\gamma}'' \equiv \frac{\gamma' Z - \gamma D(\gamma, \omega)}{Z - D(\gamma, \omega)}$$

for the terms in the denominator). One should note that the main difference between equations (25) and (27) is the fact that in the latter we assign one of the weights to a value of $\omega(\beta')$ which is the optimal weight for storing the maximal number of vectors with the ratio β' *and not* for the currently examined ratio (γ). Another main difference is that in equation (27) in the denominator we only consider all the *linearly separable* functions instead of the entire set of *possible functions* considered in equation (25).

Equation (27) can be solved in an iterative manner, when each iteration is based on probabilities computed previously for lower Z values, to obtain exact solutions. However, such an iterative computation is time consuming and we will, therefore, present two simple approximations which yield good results.

3.1. $\Phi^{\omega}(\beta'', Z) = \delta(\beta, \beta')$

Assuming that most of the functions with a ratio β are generated by a selection of $A(\beta')$ vectors where $\beta' = \beta$, we can use this approximation which results in a very simple form for equations (25) and (26).

$$P(\beta, B) = \text{Min}\{1, 2^{A(\beta)}/C_B^{\beta B}\} \tag{28}$$

$$LS(N) = \sum_{\beta} \text{Min}\{C_B^{\beta B}, 2^{A(\beta)}\}. \tag{29}$$

This approximation gives good results in comparison with the theoretical values [14, 17] and with the results obtained by the well known approximation [17] $LS(N) = 2\sum_{i=0}^{2^N} C_{2^N-1}^{i}$. The computed results for several special cases are shown in figures 5 and 6, table 2 (for the probabilities) and table 1 (for the number of linearly separable binary functions).

Table 1. The calculated number of linearly separable functions for the various approximations in comparison with the theoretical values for the $N = 1, 2, \ldots, 8$-dimensional space.

	$LS(N)$			
N	Theoretical	Approx 1	Approx 2	$2\sum_{i=0}^{N} C_{2^N-1}^{i}$
1	4	4	4	4
2	14	14	11	14
3	104	122	108	128
4	1882	1618	1706	3882
5	94 572	65 506	69 590	412 736
6	15 028 134	20 376 610	20 402 059	151 223 522
7	8 378 070 864	51 603 788 482	47 980 317 066	$189\,581\times10^{6}$
8	17 561 539 552 946	153 028 631 553 201 152		$114\,419\,417\times10^{9}$

Moreover, one can easily use this approximation to compute the number of linearly separable functions *with different restrictions*. Table 3 presents an approximation for the number of linearly separable functions for a single-layer perceptron with one weight ω fixed in several values, computed via equations (29) and (19). One should note that there is no other simple approximation to compute these values with certain weight restrictions.

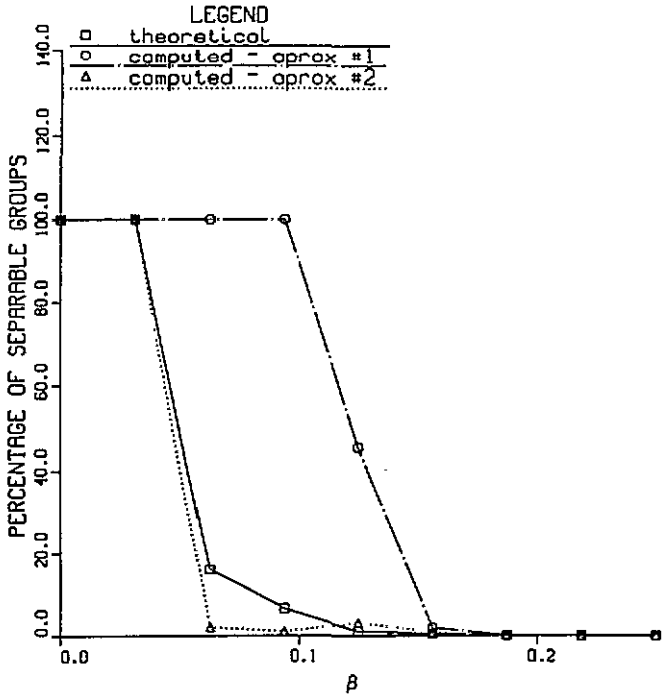

Figure 5. The probability for selecting a linearly separable function with respect to the ratio β:$N = 5$, linear scale.

3.2. $\Phi^{\omega}(\gamma'', Z - D) = \delta(\gamma, \beta')$

Refining the results, one can use this finer approximation, based on the assumption that the most probable ratio for a linearly separable function given a previously defined weight $\omega(\beta')$ and the limited set of the $Z - D$ remaining vectors in equation (27), is the one related to the enforced weight $\omega(\beta')$, i.e. the ratio β'. Using this assumption we can obtain approximated values for the various probabilities $\Phi^{\omega}(\beta'', Z)$ of the form

$$\Phi^{\omega}(\beta'', Z) = \frac{\operatorname{Min}\{C_Z^{\beta''Z}, \operatorname{Min}\{C_Z^{\tilde{\gamma}Z}, 2^{D(\tilde{\gamma},\omega)}\}\}}{\sum_{\gamma'=0}^{1} \operatorname{Min}\{C_Z^{\gamma'Z}, \operatorname{Min}\{C_Z^{\hat{\gamma}Z}, 2^{D(\hat{\gamma},\omega)}\}\}} \tag{30}$$

where all of the notations are similar to those used previously except $\tilde{\gamma}$ and $\hat{\gamma}$ which represent the ratio for which the approximation holds, i.e. for which the following equalities hold (examining the definitions for both γ'' and $\hat{\gamma}''$)

$$\beta' = (\beta'' Z - \tilde{\gamma} D(\tilde{\gamma}, \omega))/(Z - D(\tilde{\gamma}, \omega)) \tag{31}$$

$$\beta' = (\gamma' Z - \hat{\gamma} D(\hat{\gamma}, \omega))\ Z - D(\hat{\gamma}, \omega)). \tag{32}$$

Computing the various values for the probabilities $\Phi^{\omega}(\beta'', Z)$ using equation (30), and inserting them into equations (25) and (26) one obtains a better approximation for the probabilities $P(\beta, B)$ as well as for the number of linearly separable functions $LS(N)$. The computed results for this approximation for several special cases are shown in figures 5 and

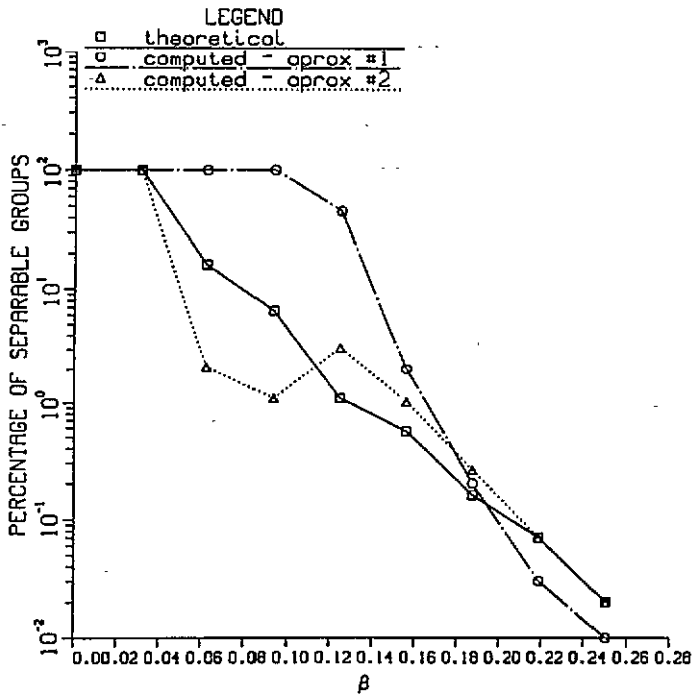

Figure 6. The probability for selecting a linearly separable function with respect to the ratio $\beta{:}N = 5$, logarithmic scale.

6, tables 1 and 2, showing superiority over the previous approximation as well as the well known approximation $LS(N) = 2\sum_{i=0}^{2N} C_{2^N-1}^{i}$.

Computing the probabilities via equation (30) requires some attention in order to deal with non-valid ratios that may occur, with the need for discretization as well as with the distortions resulting from the relatively small N numbers computed in comparison with the large N assumption used in our derivation.

To deal with the first problem we define $\Phi^{\omega}(\beta'', Z) \equiv 0$ for all $\beta'' > 1, \beta'' < 0$ and $\Phi^{\omega}(\gamma'', Z - D) \equiv 0$ for all $\gamma'' > 1, \gamma'' < 0$ (non-valid ratios). In addition we define $\Phi^{\omega}(\beta'', 0) \equiv 1$ and $\Phi^{\omega}(\gamma'', 0) \equiv 1$ for every valid ratio β', β'', γ and γ'' (i.e. when *all* vectors were previously stored).

Obviously, dealing with small N numbers all of those mentioned summations take a discrete form and therefore some discretization should be carried out. Since the values obtained for small problems are lower than the theoretical values we fixed the capacity values of $A(\beta')$ and $D(\gamma)$ to be the nearest higher integer in our simulations†. In addition we fixed the capacity $A(\beta', N = 1)$ to a value of 2 (the true value) instead of 0, to overcome the distortion resulting from the use of low values of N.

4. Perceptron convergence properties and the minimal trajectory

Most of the training methods that exist for recurrent neural networks (RNN) [5–8] as well as for feed-forward (FF) nets [9] are based on direct modification of the weight matrix,

† Obviously, one cannot store a fraction of a vector, and the discrete summation requires integer values.

Table 2. The computed probabilities of selecting a linearly separable function with a certain ratio β for a certain number of input neurons $N = 2, 3, 4, 5$ in comparison with the theoretical values.

		The probability to select a linearly separable function (%)		
N	β	Theoretical	Approx 1	Approx 2
2	1.0	100	100	100
	0.75	100	100	100
	0.5	66.7	66.7	66.7
3	1.0	100	100	100
	0.875	100	100	100
	0.75	42.9	100	62.2
	0.625	42.9	28.6	31.1
	0.5	20	22.9	43.3
4	1.0	100	100	100
	0.9375	100	100	100
	0.875	26.7	100	12.5
	0.8125	17.1	45.7	6.6
	0.75	4.8	7.0	5.1
	0.6875	4.8	2.9	4.1
	0.625	2.4	0.8	2.6
	0.5625	2.2	0.6	1.7
	0.5	0.8	0.5	1.6
5	1.0	100	100	100
	0.96875	100	100	100
	0.9375	16.1	100	2.1
	0.90625	6.5	100	1.1
	0.875	1.1	45.5	3.1
	0.84375	0.56	2.0	1.02
	0.8125	0.16	0.2	0.26
	0.78125	0.07	0.03	0.07
	0.75	0.02	0.01	0.02

Table 3. The calculated number of linearly separable functions for various selections of ω computed for the $N = 1, 2, \ldots, 8$ dimensional space.

	$LS(N, \omega)$ Approx 1			
N	$\omega = 0.2$	$\omega = 0.4$	$\omega = 0.6$	$\omega = 0.8$
1	4	4	4	4
2	14	14	12	8
3	122	114	74	44
4	1362	1522	1282	478
5	16674	35682	60610	16874
6	209986	1437378	12927714	3818034
7	2666626	75087746	12905599810	29968743698
8	34125835	3720781088	55645956440669	152863488002853888

as derived from a gradient descent procedure which is designed to decrease a given cost function. However, such cost functions are based exclusively on the set of output vectors, while the weight matrices should solve the perceptron problem common to *all*

system representations in the various time steps (for RNN) or in the various layers (for FF nets). This restriction, usually disregarded by existing training methods, might result in a non-contributing weight modification, designed to improve the current stable output vector while deforming the trajectory (we define a trajectory as the set of successive system representations produced by the system in successive time steps, for RNN, or in successive layers, for FF, in response to a presentation of a certain input vector).

Only a few algorithms for FF nets [10–12] as well as for RNN [13–15] tackle the training via modification of the internal system representations as an intermediate step towards weight matrix modifications. Selecting a proper set of system representations (or trajectory), that can be implemented by a set of perceptrons (or a single perceptron for RNN), would make the multiple-layer FF net training problem as well as the RNN training problem a simple multiple perceptron problem.

The search for a proper set of system representations (or trajectory) is aimed at producing a set of binary representations which is the most probable to be solved by the perceptron learning rule, i.e. to produce a feasible weight matrix for the entire set of representations†.

In the previous section we showed explicitly how increasing the number of vectors in the irregular group (a lower β if $\beta > 0.5$, a higher β if $\beta < 0.5$) increases the number of linearly separable functions that can be formed by randomly selected group members in the N-dimensional space. However, together with the increment in the number of linearly separable cases for a selected number of irregular group members there is a *significant* increment in the number of all possible functions that can be formed from the same ensemble and a similar ratio β. The ratio between the number of linearly separable functions that can be constructed from a random choice of a certain number of irregular vectors and the number of all possible functions constructed by that random choice defines the probability for choosing a linearly separable set of vectors given the ratio β between regular and irregular vectors. In the previous section we showed that the probability of obtaining a linearly separable set of vectors deteriorates rapidly with an increment in the number of irregular group members.

The moral of this argument is very simple. We should minimize the number of irregular vectors in each and every one of the perceptrons, representing binary representations of two successive time steps (or layers). This will increase the probability of obtaining linearly separable pairs of input–output representations to be implemented via the perceptron learning rule [16]. It is interesting to note that constructive algorithms like tiling [2] and upstart [3] basically work along these guidelines, enhancing the input–output correlation with each additional layer, keeping as many correlated input–output vector relations as possible.

Since the binary input vectors as well as the desired binary output vector representations have been previously given we search for a proper trajectory for connecting them with a minimal number of 'irregulars' in the representations of the various time steps (or layers). This trajectory is the *minimal trajectory* in state space connecting input–output representations. Two successive representations of such a trajectory, related to two successive time steps (or layers) and a certain input vector are almost identical differing statistically by

$$\frac{\sum_{p=1}^{P}\sum_{j=1}^{N}|v_j^{p,\text{in}}-\tau_j^{p,\text{out}}|}{2TNP} \tag{33}$$

† It should be emphasized that each valid trajectory connecting the input–output representations is constructed from a set of representations that are *linearly separable* for every two successive time steps (for RNN) or layers (for FF nets), and thus can be obtained by utilizing the perceptron learning rule.

bits, where $v^{\text{in},p}$ and $\tau^{p,\text{out}}$ are the binary input and desired output vectors, related to pattern p; T, N, and P are the number of time steps (or layers for FF nets) prior to stabilization, the total number of neurons and total number of patterns respectively.

Practically, this description defines *a set* of minimal binary trajectories from which one should select the most appropriate one for solving the given problem. Such a trajectory (in the RNN case) requires *a single bit flip along the entire trajectory* for each inconsistency between input and desired output neurons, which is the minimal number of bit flips possible. Such a trajectory has therefore statistically, with the maximal identity between representations, related to successive time steps (or layers), or the minimal number of vectors in the group of irregulars; regular in this sense represents vectors for which the output and the input representations are identical†. An explicit training algorithm aimed at searching for a proper minimal trajectory for RNN was presented in a previous paper [14] constructing a method for system representations and weight matrix modifications for solving certain training problem. This algorithm was examined via extensive computer simulations for some toy problems and was found to be rapid and reliable in comparison with existing training algorithms.

5. Conclusion

In this paper we have computed the capacity of the single-layer perceptron as a function of a certain input–output correlation. This correlation divides the entire set of binary vectors to be stored for each function into two groups; one for which the output neuron state equals a certain input neuron state and a second for which the output neuron state negates the state of the same input neuron. The ratio between the number of vectors in one of these groups and the entire ensemble is defined as β. We have demonstrated via statistical mechanics methods that the single-layer perceptron capacity strongly depends on this ratio—β.

Next, we used these results to compute the probability of finding a perceptron solution for a randomly selected function with a certain ratio β, giving us insight into the type of internal representation one should be looking for in order to maximize the probability of realizing them in the multi-layer perceptron/RNN framework. The results of this computation are in agreement with the theoretical figures (computed for small systems). Using these probabilities, we also computed the overall number of linearly separable functions existing in N-dimensional space (i.e. for binary vectors of N dimensions). These results, showing consistency with the theoretical values for two different approximations used, were found to be superior to a commonly used approximation [17]. In addition the general expression found extends our capability of estimating the number of linearly separable functions with various restrictions which common approximations cannot tackle.

Using the computed probabilities we explained one incentive for constructing training algorithms based on a search for the minimal trajectory for both FF nets and RNN, defining the system representations as an intermediate step towards weight matrix modifications. These incentives are a direct result of the calculation aimed at estimating the probability of finding a linearly separable function given a vector ratio β, which corresponds to the probability of implementing a set of internal representations with a certain ratio.

† Note that the continuous form of the minimal trajectory minimizes a cost function similar to the physical action integral $\int_0^T (\mathrm{d}v^p/\mathrm{d}t)^2\,\mathrm{d}t$ in the absence of potential. The discrete equivalent form of this cost function is $\sum_{t=0}^{T}\sum_{p=1}^{P}|v^p(t+1)-v^p(t)|^2$ where p is the index of the various patterns and t represents the various time steps (or layers). Hence, the said trajectory obeys a type of minimal action principle of a discrete form.

In addition, these calculations give us the basic tools for analysing constructive training algorithms based upon input–output correlation enhancement. These probabilities might enable us to estimate the capabilities of these algorithms giving fruitful ground for further research.

Acknowledgment

The author would like to thank D J Wallace for reading the manuscript as well as for helpful comments.

References

[1] Gardner E 1987 The space of interactions in neural network models *J. Phys. A: Math. Gen.* **21** 257–70
[2] Mézard M and Nadal J-P 1989 Learning in feed-forward layered networks: the tiling algorithm *J. Phys. A: Math. Gen.* **22** 2191–204
[3] Frean F 1990 The upstart algorithm: a method for constructing and training feedforward neural networks *Neural Comput.* **2** 198–209
[4] Nadal J-P 1990 On the storage capacity with sign constrained synaptic couplings *Network* **1** 463–6
[5] Pineda F J 1987 Generalization of backpropagation to recurrent neural network *Phys. Rev. Lett.* **18** 2229–32
[6] Almeida L B 1987 A learning rule for asynchronous perceptrons with feed-back in a combinatorial environment *Proc. IEEE First Int. Conf. Neural Networks II* pp 609–18
[7] Hinton G E, Sejnowski T R and Ackley D H 1984 Boltzmann machines: constraint satisfaction networks that learn *Carnegie-Mellon Technical Report* CMU-CS-84-119 (Pittsburgh: PA)
[8] Williams R J and Zipser D 1989 A learning algorithm for continually running fully recurrent neural networks *Neural Comput.* **1** 270–80
[9] Rumelhart D E and McClelland J L 1986 *Parallel Distributed Processing* vol 1 (Cambridge, MA: MIT)
[10] Saad D and Marom E 1990 Learning by choice of internal representations—an energy minimization approach *Complex Sys.* **4** 107–18
[11] Grossman T, Meir R and Domany E 1989 Learning by choice of internal representations *Complex Sys.* **2** 555–75
[12] Nabutovsky D and Grossman T 1990 Learning by CHIR without storing internal representations *Complex Sys.* **4** 519–42
[13] Saad D 1992 Training recurrent neural network via trajectory modification *Complex Sys.* **6** 213–36
[14] Saad D 1992 Training recurrent neural networks—the minimal trajectory algorithm *Int. J. Neural Sys.* **3** 83–101
[15] Rohwer R 1990 The moving targets training algorithm *Neural Information Processing Systems (Denver, 1989)* vol 2, ed D S Touretzky (San Mateo: Morgan Kaufmann) pp 558–65
[16] Minsky M L and Papert S A 1988 *Perceptrons* 3rd edn (Cambridge, MA: MIT)
[17] Muroga S 1971 *Threshold Logic and its Applications* (New York: Wiley) p 273
[18] Garey M R and Johnson D S 1979 *Computers and Intractability—A Guide to the Theory of NP-completeness* (San Francisco: Freeman) pp 167–70, 245