A Convergence Theorem for Sequential Learning in Two Layer Perceptrons

Mario Marchand*, Mostefa Golea

Department of Physics, University of Ottawa, 34 G. Glinski, Ottawa, Canada K1N-6N5

Pál Ruján †

Institut für Festkörperforschung der Kernforschungsanlage

Jülich, Postfach 1913, D-5170 Jülich, Federal Republic of Germany

PACS. 02.70 - Computational techniques.

PACS. 87.10 - General, theoretical, and mathematical biophysics

(inc. logic of biosystems, cybernetics and bionics).

PACS. 89.70 - Information science.

Accepted by Europhysics Lett. ; December 19, 1989

Abstract

We consider a Perceptron with N_i input units, one output and a yet unspecified number of hidden units. This Perceptron must be able to learn a given but arbitrary set of input-output examples. By sequential learning we mean that groups of patterns, pertaining to the same class, are sequentially separated from the rest by successively adding hidden units until the remaining patterns are all in the same class. We prove that the internal representations obtained by such procedures are linearly separable. Preliminary numerical tests of an algorithm implementing these ideas are presented and compare favourably with results of other growth algorithms.

1 Introduction

Feed-forward layered neural networks[1] are perhaps the simplest neuro-computational devices able to implement any possible association between pairs of input-output patterns, provided that enough hidden units are present[2]. For these networks, learning procedures

*E-mail: MMMSJ@UOTTAWA.BITNET

†On leave from Institute for Theoretical Physics, Eötvös University, Budapest, Hungary. e-mail: Earn/bitnet IFF162@DJUKFA11

such as back-propagation[1, 3] have been found to be computationally prohibitive[4, 5, 6]. This is not surprising since recently it has been shown[7, 8] that the problem of deciding wether or not a given task (*i.e.* a set input-output patterns) can be performed by a given architecture (*i.e.* a network for which the number of units and the connectivity is given) is NP-complete. Note, however, that if we relax the constraint of a fixed architecture during learning, the problem can be trivially solved in several ways. For example, if for each pattern to learn one *allocates* one hidden unit which fires only in the presence of that particular pattern[1], the amount of computation grows only linearly with the number of patterns to store [5]. Unfortunately, this solution is equivalent to a table look-up and does not capture any of the *correlations* between the presented patterns. In fact, the present interest on neural networks is motivated by their ability to faithfully represent such correlations, assuming the training set *has* any. The network is then able to correctly classify new incoming inputs, so it has *learned*, not only *memorized*. Moreover, it has been shown both numerically [5] and theoretically [9, 10], that this *generalization* property is most pronounced for a minimal network. (*ie.* a network with a minimum number of neurons and connections). Hence, contrary to standard approaches as back-propagation which try to *load* a given task into a given architecture, we study here a new class of learning procedures which *build* a network able to perform a given task. Since the task of finding the absolute minimal architecture representing a given training set is also NP-hard[8], we suggest *heuristic* approaches of building networks which are close to the minimal one but whose learning times are acceptable.

Very recently, two different such heuristic approaches[11, 12] have been proposed. The class of learning procedures introduced here differs in several respects from the previous ones. For the Tiling[11] strategy one generally obtains several layers before convergence and, in each layer, there exists two kinds of hidden units: a master unit obtained by minimizing an error and ancillary units obtained in order to have faithful classes. In our approach, only one type of unit is necessary and we do not minimize any error. Furthermore, our convergence theorem is quite different and applies already for the first layer of hidden units. Compared to Ruján and Marchand[12] the present approach is a substantial improvement. First, we now go beyond regular partitions[12] (and hence permit a larger set of architectures) by relaxing the condition which did not allow hyperplanes to intersect inside the hypercube. Second, in our new implementation we do not restrict the values of the connections to $\{-1, 0, +1\}$ but allow for any integer value. Third, instead of using an exhaustive search procedure among a limited set of solutions, we now use iteratively the Perceptron algorithm[2].

2 The sequential learning theorem

Consider a Boolean function defined on M (distinct) patterns $\{\vec{\xi}^{\mu}\}_{\mu=1,\ldots,M}$ of N_i binary input units $\xi_j^{\mu} = \pm 1, (j = 1, \ldots, N_i)$ whose value is given by the targets $\sigma^{\mu} = \pm 1$. The function is not necessarily defined on all the 2^{N_i} possible patterns so we have $M \leq 2^{N_i}$. We want to find a network made of h hidden units (each can be connected to all input units) and a single output unit (connected to each hidden unit but not to the input units) that performs the given Boolean function $\sigma^{\mu}(\vec{\xi}^{\mu})$ for all $\mu = 1, \ldots, M$. Denote by $w_{k,j}$ the connection

[1]This is sometimes called the "Grandmother neuron" solution.

from input unit j to hidden unit k and by $w_{k,0}$ the bias of hidden unit k. When pattern μ is presented on the input, the activation state of the k^{th} hidden unit, S_k^μ is given by the threshold rule:

$$S_k^\mu = \text{sgn}(\sum_{j=1}^{N_i} w_{k,j}\xi_j^\mu + w_{k,0}). \tag{1}$$

For each $\vec{\xi}^\mu$ presented to the net, the activations of all h hidden units form the vector $\vec{S}^\mu$ — the *internal representation* of pattern $\vec{\xi}^\mu$. Let u_j denote the connection from hidden unit j to the output unit and u_0 its bias. Thus, the state of the output unit, O^μ, is given by:

$$O^\mu = \text{sgn}(\sum_{j=1}^{h} u_j S_j^\mu + u_0). \tag{2}$$

The sequential procedure to build the layer of hidden units is defined as follows. Find any set of $w_{1,j}$'s such that $S_1^\mu = \sigma^\mu$ for *all* the patterns of target $\sigma^\mu = -s_1, (s_1 = \pm 1)$ *and* for a certain number $n_1 \geq 1$ of patterns for which their targets $\sigma^\mu = s_1$. Note that this is always possible because of the existence of the "Grandmother neuron" solution [5, 12]. This gives us the first hidden unit. Next, remove all the n_1 patterns and call the remaining set set of $M - n_1$ patterns the *working set*. Now, find a set of values for the $w_{2,j}$'s such that $S_2^\mu = \sigma^\mu$ for all patterns (of the working set only) of target $-s_2$ (again, $s_2 = \pm 1$), *and* for $n_2 \geq 1$ patterns of target s_2. We shall say that this hidden unit "excludes" n_2 patterns from the working set. Remove these n_2 patterns from the working set and we have now two hidden units. Repeat this procedure until only patterns of the same target remain in the working set.

Hence, the input space (formed by all the $\vec{\xi}^\mu$) has been partitioned into N_{ir} regions, each of them containing at least one $\vec{\xi}^\mu$ all of the same target σ^μ and identified by a single internal representation vector $\vec{S}^\mu$ (see Fig. 1a). The number h of hidden units and the cluster sizes $n_1, \ldots, n_h, n_{h+1}$ of sign $s_1, \ldots, s_h, s_{h+1}$ respectively, are not fixed in advance. Their determination is the task of the particular algorithm implementing the sequential learning. The $h+1$th cluster is made of remaining $\vec{\xi}^\mu$ of target $s_{h+1} = -s_h$ obtained at the end of the procedure. Note also (see Fig 1a) that the number of different internal representations $N_{ir} \geq h+1$, generalizing the result obtained for regular partitions [12], where $N_{ir} = h+1$.

The original Boolean function has been now mapped into a new function defined by the internal representations. We now prove the following remarkable theorem: **The set of internal representations obtained by any procedure satisfying the conditions of sequential learning is linearly separable**.

The proof of this theorem relies on the fact that we have build *sequentially* h hidden units, each of which is "excluding" from the working space a cluster of patterns of the same target. Hence, upon the presentation of any pattern $\vec{\xi}_\mu$ member of the kth cluster the internal representation vector on the hidden layer has the form:

$$\vec{S}^\mu = (-s_1, -s_2, \ldots, -s_{k-1}, s_k, *, \ldots, *). \tag{3}$$

where $* = \pm 1$ depending on the choice of the connections for the hidden units $k+1, \ldots, h$. This suggests that the desired output s_k of this vector can be obtained for the output unit if we choose the connections such that $u_k > \sum_{j=k+1}^{h} |u_j|$ and the bias u_0 such as to cancel

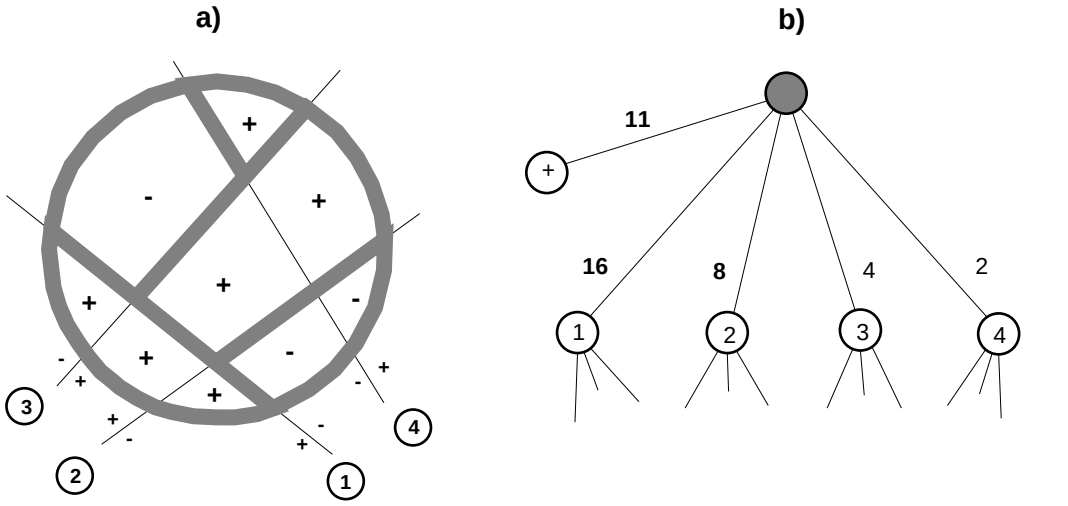

Figure 1: An example of a) the partition of the input space realized by a particular sequential algorithm and b) the weights going to the output unit given by eq. 4 and 5. In a), the large circle represents the input space and the hidden units are represented by straight lines with their outputs indicated by a + or − on each sides. In each region, all patterns have the same target wich is indicated by + or −. Here we have nine different internal representations but only five "excluded" clusters (indicated by shaded regions). The corresponding hidden units with the value of their outgoing connection are illustrated in b). The bias u_0 is connected to a unit always in the +1 state.

out the activity coming from the units $1, \ldots, k-1$. The following solution satisfies both conditions:

$$u_j = 2^{h-j+1} \quad for \quad j = 1, \ldots, h \tag{4}$$

$$u_0 = \sum_{i=1}^{h} s_i u_i - s_h \tag{5}$$

where $-s_h$ is the target of the $h+1$th cluster (see Fig. 1b). Indeed, one can verify (by using eq. 2, 3, 4 and 5) that with this solution we have $O^\mu = s_k$ upon presentation of any member of cluster k. This is a sufficient solution for any sequential learning procedure, which completes the proof. Moreover, we have found that running the perceptron algorithm on the obtained set of internal representations will generally not give this solution, where the weights increase exponentially.

3 A particular implementation

According to the sequential learning procedure, the weight vector going to a given hidden unit is chosen such as to exclude a certain number of patterns of the same class out of the working space. Since our goal is to construct as few hidden units as possible, it seems natural to choose at each step the weight vector excluding the maximal number of patterns with the same target. Such a greedy procedure is not necessarily leading to an optimal architecture but is relatively easy to implement as following.

Assume that when constructing hidden unit k, the working space has N^+ patterns of target $+1$ and N^- patterns of target -1. Our implementation tries first to find two weight vectors: one that excludes a maximal M^+ number of patterns out of the working space and another one that exclude the maximum number M^- of negative target patterns. Among these two, we choose the one that corresponds to the largest density $M^{+(-)}/N^{+(-)}$.

To find, for each hidden unit k, the weight vector that maximizes M^+, we pick a pattern $\vec{\xi}^\mu$ of positive target and we choose the weights such that the output of this unit is $+1$ for that pattern and -1 for all the others: $w_{k,j} = \xi_j^\mu$ for $j = 1, \ldots, N_i$ and $w_{k,0} = 1 - N_i$. At this point the work space consists of all -1 patterns and $\vec{\xi}^\mu$, that are properly classified by hidden unit k (*i.e.* $S_k^\mu = \sigma^\mu$) and another set of misclassified $+1$ patterns.

Next, a misclassified pattern is chosen and the Perceptron rule is used to change the weights as to classify correctly *also* this pattern. In case of failure we drop the pattern and try again with another one. After trying to include all the misclassified $+1$ patterns one has a weight vector that excludes from the working space a certain number v_1 of $+1$ patterns but, unless we have been working on a linearly separable set, there are still some misclassified $+1$ patterns left. Store this weight vector in memory, remove the properly classified $+1$ patterns from the working set and repeat the same procedure starting at a misclassified pattern. To compute what number v_2 of $+1$ patterns this weight vector excludes from the working space we need to bring back temporarily the v_1 patterns into the working space. Depending on $v_2 < v_1$ we keep only the best of the previous weight vectors. We continue, keeping always the best weight vector found, until $\sum_i v_i = M^+$. We proceed similarly to find the weight vector that maximizes M^-.

The major computational load of this procedure is the use of the perceptron algorithm to decide whether or not a given set of N patterns (in which only one pattern is misclassified) is linearly separable. We have found however that, in this case, the perceptron requires few iterations to find a solution when the set is linearly separable. For example, the maximum number of iterations found in 625 tests on $N = 1000$ for $N_i = 20$ was 105, which is much smaller than N. In addition, one can take advantage of the fact that if the set is not linearly separable, the sequence of weights given by the Perceptron rule will eventually cycle[2]. More detailed results on this problem will appear in a later publication.

4 Numerical results

Random Boolean functions. To test the robustness of the algorithm, we have generated at random 100 Boolean functions on N_i input bits. As expected, a network with one layer of hidden units was always found. For $N_i = 6$ and 8, the average number of hidden unit found was $< h >= 7.28 \pm 0.82$ and 18.3 ± 1.2 which is significantly lower than the results reported in ref. [11, 12].

Parity function. For this task the output of N_i input bits must be $+1$ if the number of $+1$ input bits is odd, and -1 otherwise. We have always found (for $N_i = 1, \ldots, 12$) only one layer of $h = N_i$ hidden units with a solution for the weights similar to those found in[1, 11, 12].

Mirror symmetry function. For this task[1], the output should be $+1$ only if the input bits are symmetric about their center. For $N_i = 2,\ 4, \ldots,\ 12$, we have always found the

optimal solution of only two hidden units. For $N_i = 6$, the weight vector going to the first hidden unit is $= (1,\ 7, 3, 1, -1, -3,\ -7))$ and the opposite (except the bias) for the weights going to the second hidden unit. Note that it is impossible to obtain this solution with the Tiling algorithm[11]. Indeed, in building the master unit, we find (by using the "pocket algorithm"[13]) eventually the weight vector that minimizes the error. Since there is only $2^{N_i/2}$ patterns of positive targets, the weight vector that minimizes the error for this task is properly classifying all the negative targets and one positive target (*i.e.* a "Grandmother" solution). This gives an error of 7 for $N_i = 6$ whereas our solution above gives an error of 28. This shows that a least error strategy, as adopted by the Tiling algorithm, is not always appropriate.

Generalization. Here we want to find out under which conditions a network, build on only parts of a function, can obtain the correct output on the remaining unseen patterns. We present here only a few results on the mirror symmetry function defined on 10 input units, a more detailed analysis will appear elsewhere. After building a network from M randomly chosen examples (among the 1024 possible choices), we run the remaining $1024 - M$ patterns. Since the number of $+1$ patterns (32) is substantially lower than the number of -1 patterns, we need to consider the results for the $+1$ patterns separately from those of the -1 patterns. The fraction of correct outputs obtained for the -1 patterns was always exceeding 95% for $M \geq 200$. It is however much more difficult to classify the $+1$ patterns properly. Averaging our results over 20 networks, we have obtained (for the $+1$ patterns) the fraction of correct outputs as: $0.13 \pm 0.13, 0.58 \pm 0.29, 0.94 \pm 0.14$ and $0.98 \pm 0.10 \text{for} M = 200, 400, 600$ and 800 respectively. Correspondingly, the average number of hidden units obtained was: $2.8 \pm 0.6, 2.8 \pm 0.8, 2.1 \pm 0.3$ and 2.0 ± 0.2. Hence, even though the standard deviation is large, the results are clear: we obtain networks with good generalization properties when the training set contains more than 600 patterns. Note also, however, that the number of hidden units needed to learned only part of the function was greater than the number (two) needed to learn the complete function! This indicates (as expected) that the greedy strategy does not always produce the minimum network.

In summary, the sequential learning theorem allows to identify a new class of learning procedures that will always build a Perceptron with only one layer of hidden units but able to execute any given Boolean function defined on a given set of examples. One particular implementation, "the maximum cluster size strategy", was successful in finding the absolute minimal network in some cases (mirror-symmetry and parity). This was not the case, however, when only part of the function was used for training. Nevertheless, networks with good generalization properties where found even in this case for the mirror-symmetry function, provided that the training set contained more than half of the total number of possible patterns.

References

[1] D. E. Rumelhart, J. L. McClelland, *Parallel Distributed Processing* Vol. 1-2, (Bradford Books, MIT Press, Cambridge , Ma., 1986).

[2] M. L. Minsky, S. Papert, *Perceptrons: an Introduction to Computational Geometry*, (exp. ed.), (MIT Press, Cambridge, MA., 1988).

[3] Y. Le Cun, CESTA-AFCET *Cognitiva*, 599-604 (1985).

[4] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *Nature* **323**, 533-536 (1986).

[5] J. Denker, D. Schwartz, B. Wittner, S. Solla, J. Hopfield, R. Howard, L. Jackel, *Complex Systems* **1**, 877-922 (1987).

[6] G. Tesauro, B. Janssens, *Complex Systems* **2** 39-44 (1988).

[7] S. Judd, *Proc. IEEE First Conference on Neural Networks, San Diego 1987*, (IEEE Cat. No. 87TH0191-7, Vol. II 685-692).

[8] A. Blum and R. L. Rivest *Proceedings of the first workshop on computational learning theory* Morgan Kaufmann, 9-18 (1988).

[9] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth *Information Processing Lett.* **24** 377-380 (1987).

[10] E. B. Baum and D. Haussler *Neural Computation* **1** 151 - 160 (1989).

[11] M. Mézard, J.P. Nadal, *J. Phys. A*, **22**, 2191-2203 (1989).

[12] P. Ruján, M. Marchand, *Complex Systems* **3**, 229–242 (1989). *Proceedings of IJCNN 1989, Washington D.C*, Vol II. 105-110.

[13] S. I. Gallant, *IEEE Proc. 8th Conf. on Pattern Recognition*, Paris (1986).