LETTER Communicated by Wolfgang Kinzel

Generalization and Selection of Examples in Feedforward Neural Networks

Leonardo Franco
Sergio A. Cannas
Facultad de Matemática, Astronomía y Física, Universidad Nacional de Córdoba, Ciudad Universitaria, (5000), Córdoba, Argentina

In this work, we study how the selection of examples affects the learning procedure in a boolean neural network and its relationship with the complexity of the function under study and its architecture. We analyze the generalization capacity for different target functions with particular architectures through an analytical calculation of the minimum number of examples needed to obtain full generalization (i.e., zero generalization error). The analysis of the training sets associated with such parameter leads us to propose a general architecture-independent criterion for selection of training examples. The criterion was checked through numerical simulations for various particular target functions with particular architectures, as well as for random target functions in a nonoverlapping receptive field perceptron. In all cases, the selection sampling criterion lead to an improvement in the generalization capacity compared with a pure random sampling. We also show that for the parity problem, one of the most used problems for testing learning algorithms, only the use of the whole set of examples ensures global learning in a depth two architecture. We show that this difficulty can be overcome by considering a tree-structured network of depth $2\log_2(N) - 1$.

1 Introduction

Feedforward neural networks have been used extensively to solve many kinds of problems, being applied in a wide range of areas covering subjects such as prediction of temporal series, structure prediction of proteins, and speech recognition. (See, e.g., Haykin, 1994, and Hertz, Krogh, & Palmer, 1991.) One of the fundamental properties making these networks useful is its capacity to learn from examples. Through synaptic modifications algorithms, the network is capable of obtaining a new structure of internal connections that is appropriate for solving a determined task.

The general underlying theory of the whole learning process is poorly understood. There are few general results, especially concerning generalization. One particular point of interest is the selection of a concise subset of examples from the whole training set as a way of improving generaliza-

tion ability. This problem has also been referred to as "active learning" or "query-based learning" by many authors. In a broad sense, these terms refer to any form of learning in which the learning algorithm has some control over the inputs used for the training.

Several approaches have been developed to derive criteria for selecting examples in different sort of networks, ranging from simple nets as perceptrons (Kinzel & Ruján, 1990) and linear classifiers (Jung & Opper, 1996) to multilayer feedforward networks with continuous outputs. Among them (see Plutowski & White, 1993, for more references) we refer to the work of Cohn (Cohn, Atlas, & Ladner, 1994; Cohn, 1996), which uses partially trained networks to determine regions of uncertainty in the environment from which the examples are selected. Plutowski and White (1993) assume that a large amount of data have been collected and work on principles for selecting a subset of those data for efficient training. Another kind of approach is that of Baum (1991), who proposed a query-based algorithm for iteratively looking for classification boundaries in the input space.

An important question related to the selection of example problems is, What is the *absolute minimum number of examples* that *ensures full generalization* in the learning procedure? By full generalization, we mean zero generalization error in the case of boolean networks or an error less than some small, positive value for the case of continuous outputs. A minimum number is the size of some particular subset of examples that contains all the information about the task or target function to be learned, and it gives a lower bound for the number of examples needed for full generalization in any selective sampling procedure (given a target function and a fixed architecture). The counterpart is the minimum average number of examples needed for generalization in a pure random sampling of the training set (Baum & Haussler, 1989). In this case the Vapnik-Chervonenkis (VC) dimension (see Haykin, 1994) appears to be a fundamental parameter for determining the generalization capacity of a fixed architecture.

Although the minimum number of examples needed for full generalization (MNEFG) is related to the intrinsic complexity of the target function, it also depends on the architecture of the student network (incorrectly designed nets might not be able to use all the information contained in the set and therefore need more examples to achieve full generalization). In this sense the MNEFG results in a more specific concept than the VC dimension, since the former is related to the combined complexity of the architecture and the specific target function, while the latter refers to arbitrary target functions. In fact, the MNEFG is an upper bound for the VC dimension.

In sections 2, 3 and 4, we analyze the learning of three different boolean problems by particular architectures using linear threshold units and obtain analytically a lower bound for the MNEFG. The idea of the method is the following. Suppose a boolean network has N inputs, a single output, and a given target function. Then the perfect learning of every one of the 2^N possible examples implies a set of constraints on the synaptic weights. In a

general case, only a subset of the 2^N constraints will be independent. Since every constraint is associated with one particular example, the size of the subset gives us the minimum number of examples to ensure the perfect learning of the 2^N examples. The analysis was carried out first for a small number of inputs and then extended to arbitrary sizes N.

Most of the analysis in this article was carried out with specific problems and architectures. Models for which a set of exact solutions and properties are known serve as standards for testing the hypothesis. The work presented here illustrates how insights about general properties of learning can be obtained from studying of such standards.

The three problems we first considered were addition of two numbers, bit shifting, and parity. For every problem, we chose an architecture for which the computability was already known. Moreover, for each of these structures, a set of exact solutions was available (see Cannas, 1995, for the first case; Franco & Cannas, 1998, for the second one; and Rumelhart & McClelland, 1986, for the last one). The three structures have N inputs, a single output (in order to make the results comparable in the three cases, we considered only the most significant bit of the output in the first two cases), and one hidden layer. Since all the problems are nonlinearly separable, the structures are optimal (i.e., minimal) in depth in all the cases. We found that the MNEFG scales polynomially with N in the first two cases and exponentially in the last one. That is, for the parity function, the MNEFG scales with the total number of examples for a depth 2 network.

The main difference between the architectures is that while in the parity case it is fully connected with N hidden units, in the other two cases we chose architectures with fewer hidden units and local receptive fields. These ad hoc modifications simplify the analytical treatment without a great loss of generality since they are based on prior knowledge of the global symmetries of the corresponding problems (Cannas, 1995; Franco & Cannas, 1998). Although they reduce the computability of the network (in the sense of the number of different target functions that the architecture is capable of implement) compared with the fully connected counterpart, these architectures are expected to be general enough to compute a large class of functions, at least those sharing the same types of global symmetries. In particular, for the addition case, the reduction becomes negligible for large values of the number of inputs N, because the number of hidden units is kept constant. Moreover, the presence of local receptive fields and a small number of hidden units can always be thought of as a particular solution of a fully connected network with N hidden units and a set of synaptic weights equal to zero. Hence, the MNEFG we obtained constitute lower bounds for the three problems considered here when implemented in a standard architecture—a fully connected depth 2 network with the same number of inputs and hidden units. Since the three problems are expected to have different complexity, the MNEFG appears to be an interesting parameter to classify the complexity of a target function. Moreover, an analysis of the

subsets of examples (or "exemplars," in Plutowski and White's terminology) that determine the MNEFG gave us a first insight into how to construct a criterion for selecting examples.

The analysis is detailed in section 5, where the criterion is tested in the nonoverlapping receptive field perceptron (Hancock, Golea, & Marchand, 1994). An analysis of the MNEFG for a particular target function in this general architecture allowed us to refine our ideas and propose a general architecture-independent criterion for selecting examples in boolean networks. This criterion was checked for random functions using numerical simulations in a small network, showing a considerable improvement in the generalization capacity compared with a pure random sampling, especially concerning the probability of full generalization.

We also present some numerical simulations for the addition and bit shifting problems in small networks.

The criterion works well in all these case but fails in the parity one. To show that the difficulty here comes from the architecture, which is not complex enough to accomplish the desired task (the depth 2 network can learn, but it cannot generalize), we study a network of depth $2\log_2 N-1$ with a tree structure, which is also capable of computing the parity function (Rumelhart & McClelland, 1986). We show that full generalization is possible with this architecture, for which the MNEFG scales polynomially with N. Moreover, we show through numerical simulations that the generalization capacity is also improved in this case with our selection criterion.

2 The Sum Problem

We study the generalization properties of a network constructed to compute the addition between two binary operands each of N bits, which gives a result of the same length N. The architecture (Cannas, 1995) is optimal in depth; it has only one hidden layer. It is composed of $2N$ binary neurons in the input layer, corresponding to the two numbers of N-bits to add, N hidden neurons and N binary neurons in the output. To simplify the analysis and allow comparisons with every function that has at least one output neuron, we study only one output bit—the one having the most significant value. The generalization to the case of N output bits is straightforward, since the learning of the most significant output bit automatically fixes the synapses shared with the least significant output bits to its correct values. The rest of the synapses can be fixed by a set of independent exemplars selected by the same procedure used for the most significant bit. The resulting network has two hidden neurons: one is fully connected to the input layer, and the other is connected to every input neuron but those corresponding to the most significant bits. Full connection also exists between the hidden layer and the output neuron. Finally, the two most significant input bits are also directly connected to the output. An example of the network for $N = 3$ is shown in Figure 1a, where only the connections related

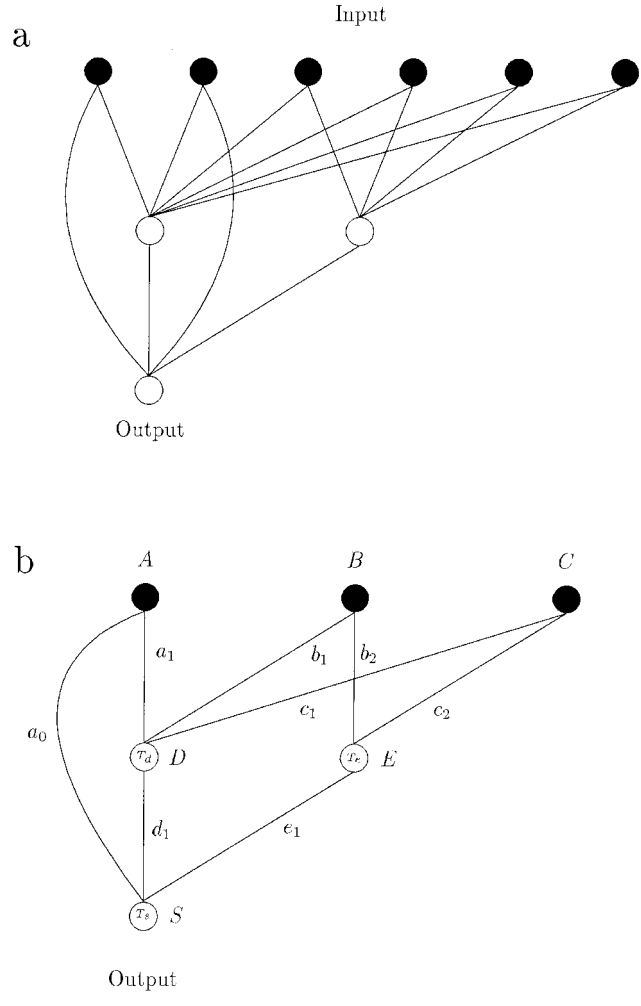

Figure 1: Network structure to compute the most significant output bit of the sum function of two three-bit numbers. (a) General structure for arbitrary synapses without constraints. (b) The synapses corresponding to input bits with the same significant value have been symmetrized in such a way that the six input bits can be replaced by three input bits (denoted by A, B, and C) taking the values {0, 1, 2}.

to the most significant bit are shown. We see that the architecture is almost fully connected, where only a set of backward connections (from left to right) has been set to zero. This simplification is based on prior knowledge

of the global left-right asymmetry of the addition, produced by the carry operation from the least (right) to the most (left) significant bits (Cannas, 1995). Hence, this architecture is expected to be general enough to compute a large class of functions, at least those sharing the left-right asymmetry.

Further simplification of the problem can be obtained by considering shared synapses. That means we impose the constraint that synapses connecting one hidden neuron with two input bits with the same significant value are always equal. With this constraint, every pair of input neurons corresponding to the same significant bit can be replaced by a single ternary neuron, which takes the values $\{0, 1, 2\}$. Hence, our final architecture contains an input layer of N ternary neurons. We start by adding two numbers of 3-bit length and then generalize the results to the addition of two N-bits numbers. A description of the final network architecture for $N = 3$ is depicted in Figure 1b.

The output bit S receives direct inputs from the two hidden neurons $D, E = 0, 1$ through synapses d_1, e_1 and from the input neuron A, through synapse a_0, computing the following function:

$$S = \theta\{a_0 A + d_1\theta[a_1 A + b_1 B + c_1 C - T_d] + e_1\theta[b_2 B + c_2 C - T_e] - T_s\}, \quad (2.1)$$

where $A, B, C = 0, 1, 2$; $\theta(x)$ is the Heaviside step function and T_α ($\alpha = d, e, s$) are the threshold parameters.

The values of the synapse a_0 and those of the thresholds T_d and T_e are restricted to be greater than zero to reduce the possible internal representations to only one.

Requiring that the network compute the full set of 27 addition examples leads to the following necessary and sufficient conditions:

$$T_d > a_1 \geq \frac{T_d}{2} \quad (2.2)$$

$$a_1 + 2b_1 \geq T_d > 2b_1 + 2c_1 \quad (2.3)$$

$$a_1 + b_1 + 2c_1 \geq T_d > a_1 + b_1 + c_1 \quad (2.4)$$

$$2b_2 \geq T_e > 2c_2 \quad (2.5)$$

$$b_2 + 2c_2 \geq T_e > b_2 + c_2 \quad (2.6)$$

$$e_1 \geq T_s > 0 \quad (2.7)$$

$$a_0 \geq T_s > 2a_0 + d_1 \quad (2.8)$$

$$2a_0 + d_1 + e_1 \geq T_s > a_0 + d_1 + e_1. \quad (2.9)$$

We will show that if the network computes a particular set of 12 examples, then the above conditions are satisfied. Therefore, the correct learning of such examples ensures full generalization with fewer than half of the total number of examples.

We will denote the examples by writing between square brackets the three input values and the correct output separated by a colon. Then:

- From the example [000:0], we obtain the right part of equation 2.7.
- From the examples [100:1] and [200:0] we obtain:

$$d_1 < -a_0 \tag{2.10}$$

  and the fulfillment of equations 2.2 and 2.8.
- From the example [020:1] we obtain:

$$d_1\theta[2b_1 - T_d] + e_1\theta[2b_2 - T_e] > T_s,$$

  which together with $T_s > 0$ and equation 2.10 implies the left part of equation 2.7 and the left part of equation 2.5.
- From the example [120:0], we obtain the left part of equation 2.3, together with the right part of equation 2.9.
- From the examples [111:1] and [112:0], we obtain equation 2.4 and also that $c_1 > 0$.
- From the examples [011:0], [012:1], and [002:0], we obtain equation 2.6 together with the right part of equation 2.5.
- From the example [222:1], we obtain the left part of equation 2.9.
- From the example [022:1], we obtain the right part of equation 2.3 and consequently the fulfillment of the complete set of equations 2.2 through 2.9.

For the most general case of an input of N bits, we can see that for each bit we increase the input in size, it is enough to add four examples to obtain full generalization. For example, for the case of $N = 4$ we take the 12 examples corresponding to the 3-bit case but converted to the new problem by adding a zero in the new right-most input place. For example, the input pattern [112:0] in the 3-bit problem now would be the input pattern [1120:0]. It is also necessary to add four new examples, two pairs of them having different outputs when we modify the new right-most input bit. For the case of $N = 4$ these two pairs are the examples {[0111:0][0112:1]; [1111:1][1112:0]}. The same procedure is repeated as we increase the number of input bits. Thus, the size of the minimal set of examples needed for generalization is equal to $4N$. Finally, for N output neurons, such a number becomes $2N(N + 1)$.

3 Bit Shifting

The bit-shifting function is a basic operation in computer circuits and also was used in modeling vision devices (see Franco & Cannas, 1998). The network structure has only one hidden layer, and since the problem is linearly

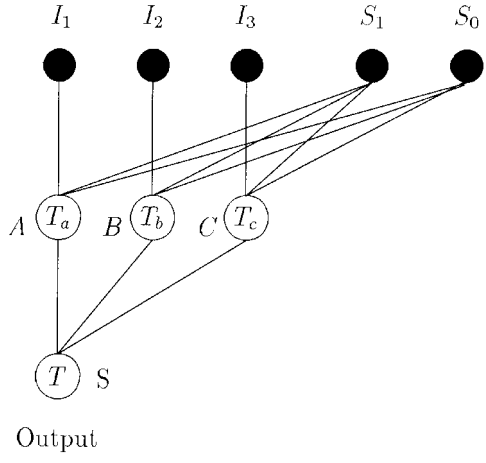

Figure 2: Network structure to compute the left-most output bit of a bit-shifting operation, where I_1, I_2, I_3 are the input bits and S_1, S_0 are the bits indicating the number of bits to shift.

inseparable, the structure is optimal in depth. The input layer has N input bits to shift plus $\log_2(N+1)$, indicating bits given the places to shift. There are N binary neurons in the hidden layer, each one connected to one input neuron and to all the indicating neurons. The problem has an output of N bits, but to simplify the study, we analyze just one output bit. We start the analysis with the particular case of having three input bits with its corresponding two indicating bits, and later generalize the results to the case of N input bits.

For the case of three input bits, the structure of the network is shown in Figure 2, where the output neuron S computes the following function:

$$S = \theta\big[J_a\theta(a_1I_1 + a_2S_1 + a_3S_0 - T_a) + J_b\theta(b_1I_2 + b_2S_1 + b_3S_0 - T_b) + J_c\theta(c_1I_3 + c_2S_1 + c_3S_0 - T_c) - T\big] \tag{3.1}$$

$I_1, I_2, I_3 = 0, 1$ being the input bits; $S_0, S_1 = 0, 1$ are the two indicating bits, J_a, J_b, J_c are the synapses between the three hidden neurons A, B, C and the output S, and a_i, b_i, c_i are the synapses between the respectively hidden neurons (A,B,C) and the input and indicating bits.

We impose the constraint that the thresholds of the hidden neurons T_a, T_b, T_c are always positive to reduce to only one the number of possible internal representations. In order to simplify the generalization to the N input bits case, we also impose that the synapses J_a, J_b, and J_c have to be greater than the threshold of the output neuron T. These constraints produce no substantial changes to the results.

Requiring that this network compute exactly the full set of 32 examples leads to the following necessary and sufficient conditions:

$$T > 0 \tag{3.2}$$
$$a_1 \geq T_a > a_1 + a_3 \tag{3.3}$$
$$a_1 + a_2 < T_a \tag{3.4}$$
$$b_1 < T_b \tag{3.5}$$
$$b_1 + b_3 \geq T_b > b_3 \tag{3.6}$$
$$b_1 + b_2 + b_3 < T_b \tag{3.7}$$
$$c_1 < T_c \tag{3.8}$$
$$c_1 + c_2 \geq T_c > c_2 \tag{3.9}$$
$$c_1 + c_2 + c_3 < T_c. \tag{3.10}$$

We denote the examples by writing between square brackets the three input values plus the two indicating bits and the correct output separated by a colon.

As in the previous section, it is easy to verify that the correct learning of the 10 examples—{[000–00:0], [100–00:1], [100–01:0], [100–10:0], [010–01:1], [010–11:0], [010–00:0], [001–00:1, [001–10:1], [001–11:0]} ensures the correct computation of the full set of examples. For instance, from example [000–00:0] we derive $T > 0$ (see equation 2.10); from example [100–00:1] we obtain the left side of equation 3.2; and so on.

Extending this result to the case of having N bits involves all the examples that have only one input bit ON and all its possible combinations of indicating bits, plus the patterns with all the input bits OFF and its combinations with the indicating bits. This procedure gives that for N input bits, $(N + 1)^2$ examples are enough to obtain full generalization.

4 The Parity Problem

The parity function is one of the most used problems for testing learning algorithms because its simple definition and great complexity given by the fact that the most similar patterns (those differing by a single bit) have different outputs (Rumelhart & McClelland, 1986; Tessauro & Janssens, 1988). The parity function has only one output neuron that indicates when it is ON that an odd number of the N input bits are ON, while it is OFF if this number is even. The simplest architecture known to compute this function using linear threshold units (l.t.u.) and having no direct input-output connections consists of a network with one hidden layer with N units, fully connected to the N input neurons.

As we did in the previous sections, we first analyze a particular case with six input neurons (see Figure 3) and then generalize the result to N input bits.

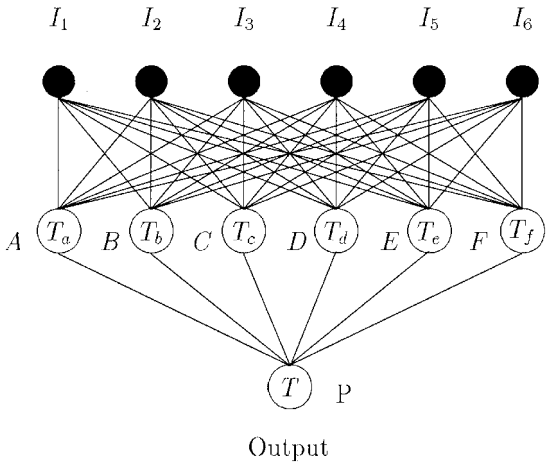

Figure 3: Network structure with one hidden layer of six fully connected neurons used to compute the 6-bit parity function.

The standard functioning of this network (see Minsky & Papert, 1969; Rumelhart & McClelland, 1986; and Hertz et al., 1991) is based on the behavior of the six hidden neurons: the number of hidden neurons that have to be ON is equal to the number of bits ON in the input. These hidden neurons are connected to the output neuron P through synapses set alternatively to positive and negative values, so the neuron P is ON when an odd number of hidden neurons are ON and is inactive when this number is even.

The functioning of the hidden neurons is achieved by the following conditions:

1. The hidden neuron A has to be ON when one or more input bits are ON; otherwise, A has to be OFF.
2. The hidden neuron B has to be ON when two or more input bits are ON; otherwise, B has to be OFF.
3. The hidden neuron C has to be ON when three or more input bits are ON; otherwise, C has to be OFF.
4. The hidden neuron D has to be ON when four or more input bits are ON; otherwise, D has to be OFF.
5. The hidden neuron E has to be ON when five or more input bits are ON; otherwise, E has to be OFF.
6. The hidden neuron F has to be ON when the six input bits are ON; otherwise, F has to be OFF.

Denoting by $P = 0, 1$ the output value, we have that P computes the following function:

$$P = \theta\left\{a\theta\left[\sum_{i=1}^{6}(a_i I_i - T_a)\right] + b\theta\left[\sum_{i=1}^{6}(b_i I_i - T_b)\right] + c\theta\left[\sum_{i=1}^{6}(c_i I_i - T_c)\right] + d\theta\left[\sum_{i=1}^{6}(d_i I_i - T_d)\right] + e\theta\left[\sum_{i=1}^{6}(e_i I_i - T_e)\right] + f\theta\left[\sum_{i=1}^{6}(f_i I_i - T_f)\right] - T\right\}, \quad (4.1)$$

where $I_i = 0, 1$ $(i = 1, \ldots, 6)$ denote the state of the input neurons.

From the conditions 1–6 we obtain the following set of inequalities for the synapses values:

- From condition 1 we obtain that:

$$a_i \geq T_a \quad \forall i. \quad (4.2)$$

- From condition 2 we obtain that:

$$b_i + b_j < T_b \quad \forall\, i, j. \quad (4.3)$$

- From condition 3 we obtain that:

$$c_i + c_j + c_k \geq T_c \quad \forall\, i, j, k. \quad (4.4)$$

- From condition 4 we obtain that:

$$d_i + d_j + d_k + d_l < T_d \quad \forall\, i, j, k, l \quad (4.5)$$

- From condition 5 we obtain that:

$$e_i + e_j + e_k + e_l + e_m \geq T_e \quad \forall\, i, j, k, l, m. \quad (4.6)$$

- From condition 6 we obtain that:

$$f_i + f_j + f_k + f_l + f_m + f_n < T_f \quad \forall\, i, j, k, l, m, n \quad (4.7)$$

(a, b, c, d, e, f).

Fixing the values of the thresholds $T_a, T_b, T_c, T_d, T_e, T_f$, equations 4.2 through 4.7 lead to a set of $2^6 - 1$ independent inequalities for the synapses' parameters $\{a_i\}, \{b_i\}, \ldots, \{f_i\}$. On the other hand, the total number of examples is 2^6, and as the learning of any example ensures only the fulfillment of one of the $2^6 - 1$ inequalities, only imposing the correct learning of the full set of examples except the simplest one [000000:0], which determines

Table 1: Some Features of the Networks used to Compute the Addition of Two Numbers, Bit-Shifting, and the N-Bit Parity Functions.

	Addition	Bit Shifting	Parity
Number of synapses	$2N + 4$	$N(2 + \log_2(N + 1))$	$N^2 + N$
Total number of examples	3^N	$2^N(N + 1)$	2^N
MNEFG	$4N$	$(N + 1)^2$	$\mathcal{O}(2^N)$

the sign of the output threshold, ensures the fulfillment of inequalities (see equations 4.2–4.7). Hence, we cannot guarantee generalization with any particular subset of examples.

Finally, the generalization of this result to a fully connected network with N inputs and N hidden neurons is straightforward, because in this case we have 2^N examples and $2^N - 1$ inequalities of the type of equations 4.2 through 4.7).

The previous analysis was based on one particular internal representation. These network admits other internal representations like that used in Franco and Cannas (1998) for the construction of a network for the addition problem. Along the same lines as before, it can be shown that in this case, the minimum number of examples, although less than $2^N - 1$, is still of order $\mathcal{O}(2^N)$. These results suggest that in this simple architecture for the parity problem (only one hidden layer, fully connected), the MNEFG could be always exponential in the number of inputs.

5 Examples

In the previous section we analyzed three problems that are expected to have different degrees of complexity, being the parity the "hardest" one. The results are summarized in Table 1. The MNEFG we obtained in the previous section can be considered the lower bounds for the corresponding problems in a fixed standard architecture—a fully connected depth 2 network with the same number of inputs and hidden units. In this sense, the results are comparable for the three problems, suggesting the MNEFG as a possible parameter for classifying complexity.

A careful analysis of the exemplars we found for the sum and bit-shifting problems shows that almost all of them can be grouped into pairs whose inputs differ in a single bit and whose output is different. In other words, such pairs are the closest ones (considering the Hamming distance between the input patterns) located at both sides of a classification boundary. This appears as a general and simple criterion for the selection of examples.

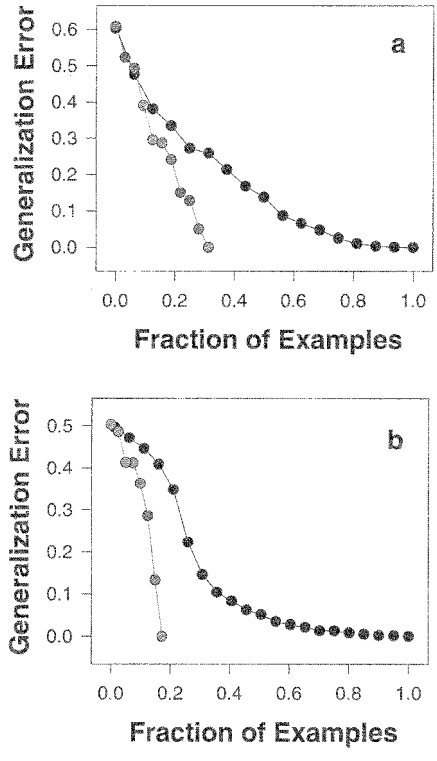

Figure 4: Comparison between the generalization error from selective sampling of examples (lower curves) and a pure random one (upper curves) for (a) a bit-shifting network with three input bits and (b) a network constructed to compute the addition of two numbers of four bits length. The pure random selection was done without repeating the examples.

In Figure 4 we show numerical simulations of the generalization error as a function of the fraction of examples in the training set using random epochs of examples selected with the above-described criterion. The results are compared with the corresponding ones for a random selection over the full set of examples obtained through numerical simulations. In Figure 4a the comparison is done for a bit-shifting network with three input bits (see the architecture of Figure 2) and in Figure 4b, the results for a network computing the addition of two numbers of four bits length (see the architecture of Figure 1) is shown. All the simulations in this work were done using simulated annealing as the learning algorithm, and the results were averaged over different random epochs and random initial distributions of the synaptic weights. Typical sample sizes run from 25 to 100.

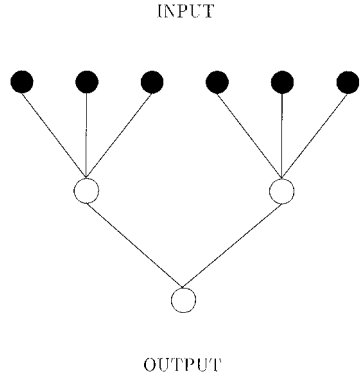

Figure 5: Nonoverlapping receptive field perceptron (NORFP) with $N = 6$ and two hidden neurons.

In order to check how the above criterion works in a general architecture, we consider another well-studied depth 2 one: the nonoverlapping receptive field perceptron (NORFP), whose structure consists essentially of many simple perceptrons connected to a single-output neuron. One can think of these architectures as networks of "decoupled" perceptrons, which in terms of architectural complexity lie between the single perceptrons and the fully connected feedforward nets. In this sense, this type of architecture has been a natural candidate for studying the learning and generalization properties started in perceptrons with no hidden units (Copelli & Caticha, 1995). An example with two hidden units and $N = 6$ inputs is depicted in Figure 5. Some properties and computational capabilities of this kind of structure can be found in Hancock et al. (1994) and Priel, Blatt, Grossman, Domany, and Kanter (1994).

First, from the many functions that these architectures can compute (Priel et al., 1994), we selected a target that is simple to analyze and whose computability is ensured. Consider the case of even N and divide the input into two groups of $N/2$ bits; then the output of the network should be ON only if there are two or more input neurons activated in each group at the same time, and OFF otherwise. We will call this function F_2. It is easy to see that this function is solvable by a NORFP with two hidden units, like the one shown in Figure 5. The problem of determining whether two or more input bits are ON in a set of $N/2$ inputs is clearly linearly separable; hence, the problem can be solved by two simple perceptrons coupled to a third one that performs a boolean AND function.

From the definition of the target function, it is simple to see the number of examples that can be grouped into pairs with only one input bit different and different output scales as $\mathcal{O}(N^2\, 2^{N/2})$ for large N. Hence, it is not expected that a random sampling of examples based on this criterion

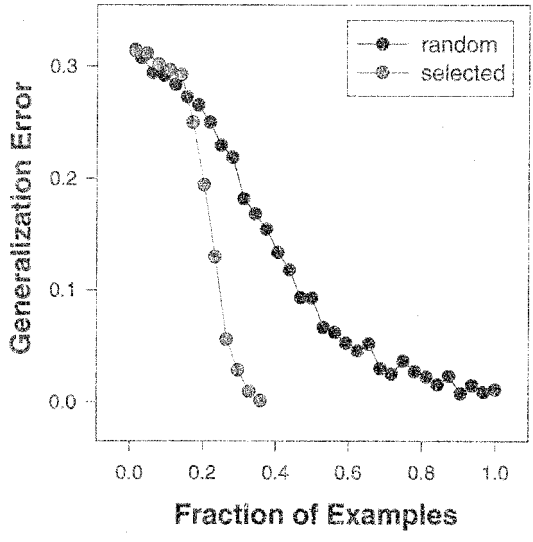

Figure 6: Comparison between the generalization error from selective sampling of examples and a pure random one in the learning of the F_2 function (see the text for a definition) by a NORFP with $N = 6$ (see Figure 5).

can improve the generalization capacity of the net, compared with a pure random sampling for this function. We verified this assumption through numerical simulations. However, not all these examples are needed to ensure full generalization. Following the same method of the previous sections, it can be seen that the set that determines MNEFG consists of all the following pairs of examples: one example with two input bits ON in a group and only one in the other group and the example having the same input bits ON in the first group plus another bit ON in the group that had only one. For example, in the case of Figure 5, one possible pair of examples is [110–100:0] and [110–110:1]. A simple counting shows that the MNEFG in this case scales as $\mathcal{O}(N^2)$ for large N. In Figure 6 we plot the generalization error obtained through numerical simulations as a function of the fraction of examples in the training set constructed with the last criterion for the network of Figure 5, and also the same curve is shown for pure random sampling of examples.

This last result led us to propose the following general criterion for selecting examples in boolean networks: select randomly only pairs of examples differing in one input bit with different outputs, but select with higher probability those having fewer active input bits. Related criteria have been implemented in other methods of active learning (Baum & Haussler, 1989; Kinzel & Ruján, 1990).

We checked this last criterion in the NORFP of Figure 5 for random target functions. In order to ensure that only computable functions were considered, we constructed a priori a set of testing functions by a random generation of synaptic weights. The results were averaged over the different functions and different random initial conditions. Sample sizes ran from 100 to 1000. The examples were selected with a probability proportional to $\propto 1/n_a^{\beta}$, where n_a is the number of active bits of the input and $\beta > 0$ is some arbitrary exponent. The results are shown in Figure 7 compared with the corresponding ones for a pure random sampling of examples. It is worth stressing that in boolean networks, full generalization can always be achieved with a large enough set of examples. Although we see improvement in the mean generalization error with our criterion (see Figure 7a), the behavior of the probability of full generalization (see Figure 7b) is much more impressive, showing a finite probability (about 10%) even with just 10% of the examples and reaching a value higher than 80% for half of the examples.

A similar test of the criterion was carried out for the addition and bit-shifting problems with the architectures of Figures 1 and 2, respectively. The results are qualitatively similar to those of the NORFP, especially concerning the probability of full generalization. We have seen that our criterion is closely related to the behavior of the MNEFG. In the case of the parity function with the standard architecture, we have shown that the MNEFG scales with the total number of possible examples, and therefore the selection criterion does not work. This example illustrates how the generalization capacity of a network is a result of the interplay between the complexity of the target function and the architecture. The previous structure is the simplest depth 2 one that can solve the parity function, and it has a very poor generalization capacity. More complex nontrivial architectures can be constructed by increasing its depth. A good example is that proposed by Rumelhart and McClelland (1986), based on the following simple principle. Suppose that $N = 2^m$ with $m > 0$ integer (generalization to other cases are straightforward). Then group the input units into 2^{m-1} pairs, and compute the parity of every pair by a standard depth 2 network (with two hidden units). This procedure can be repeated recursively, giving a tree-structured network of depth $2\log_2(N) - 1$, which solves the parity of the N inputs. An example for $N = 4$ is shown in Figure 8. As in the example of sum and bit shifting, this network incorporates a previous knowledge of a global symmetry of the target function—let us say, the "self-similarity" property of the parity.

We checked our criterion numerically in the small network of Figure 8. The results for the generalization error compared with those corresponding to a pure random sampling are shown in Figure 9. Following the same procedure of the previous sections, it can be shown that the MNEFG in this case scales as $\mathcal{O}(N^2)$. This is consistent with the observed result that even for a pure random sampling, this architecture always achieves full generalization with a large enough training set.

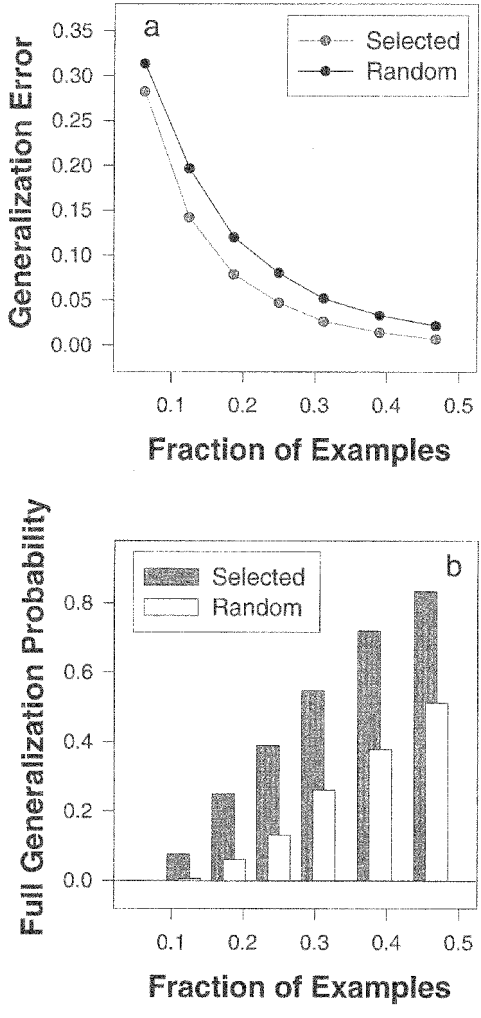

Figure 7: Selective versus pure random sampling of examples in the learning of random target functions by a NORFP with $N = 6$ (see Figure 5). (a) Average generalization error versus fraction of the total number of examples in the training set. (b) Probability of full generalization (zero generalization error).

Finally, we estimated the different average CPU times used by the learning procedure in a Pentium II (300 MHz) personal computer by three particular target functions: (1) bit-shifting ($N = 5$) with the architecture of Figure 2; (2) F_2 function ($N = 6$) in a NORFP (see Figure 5), and (3) parity with the architecture of Figure 8. The results are summarized in Table 2.

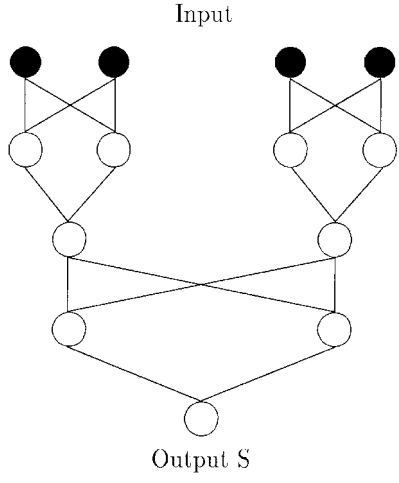

Figure 8: Tree-structured network that solves the parity function for $N = 4$.

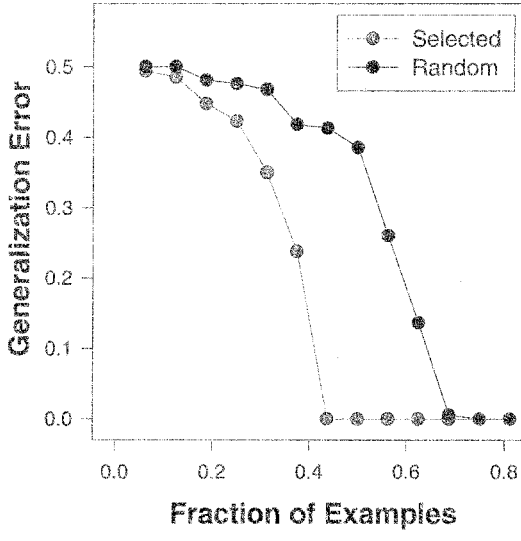

Figure 9: Comparison between the generalization error from selective sampling of examples and a pure random one in the learning of the parity by a tree-structured network with $N = 4$ (see Figure 8).

Table 2: Average CPU Time Used by the Different Sampling Procedures.

	Bit Shifting			F_2			Parity		
	Selected	Random		Selected	Random		Selected	Random	
Fraction of examples	0.28	0.28	0.81	0.4	0.4	1	0.43	0.43	0.68
Generalization error	0.0	0.26	0.02	0.0	0.15	0.02	0.0	0.4	0.0
CPU time (sec)	7.0	0.2	1.1	7.7	9.0	93.3	160	8.8	332

In all the cases, we calculated first the average CPU time needed to obtain zero generalization error with the selection sampling procedure, for a fraction of training examples corresponding to the threshold value at which the learning curve reaches zero the first time (see Figures 4a, 6, and 9). In order to compare with the performance obtained by a purely random sampling, we calculated in this case two different CPU times. First, we calculated for each target function the average time needed to converge when using the same fraction of training examples as in the selection case (which gives a larger generalization error in the random sampling than in the selective one). Second, we calculated the average time needed to obtain the minimum generalization error for each function with random sampling (which involves a larger training set, as in the previous situation).

We see that in the worst case (bit shifting), the average time needed to reach the minimum error with the selection procedure was about seven times the corresponding one for the random procedure. However, it is worth noting that in the first case, full generalization is always achieved (i.e., with probability one) only with a fraction of examples, while in the second case, even with all the examples, the probability of learning is less than one. Hence, a greater generalization capacity is obtained with the selection sampling in this case at the cost of computational time. In the other two cases, the time performance of the selection sampling compares well with the random one.

6 Conclusions and Discussion

We analyzed the generalization capacity of several boolean neural networks for different target functions. Our analysis was based on the analytic estimation of the MNEFG and its scaling properties with the number of inputs N.

A close examination of the subsets of examples that determine the MNEFG in the different cases led us to propose a simple selection criterion: first choose pairs of examples with closest inputs in terms of Hamming distance and with different outputs; examples are then chosen randomly within the data set formed in such a way, but with higher probability for those examples with fewer active input bits. This criterion is architecture independent.

However, a set of examples constructed with such a criterion is not always enough to ensure full generalization, as we have shown, for instance, for the parity. Here is where the architecture comes into play. Our results suggest that the criterion works when the MNEFG scales polynomially with N, and this seems to happen only with local receptive fields, that is, when the network is not fully connected. This effect is possibly related to the existence, in a fully connected network, of multiple internal representations for a given target function (Boers, Kuiper, Happel, & Sprinkhuizen-Kuyper, 1993). In this case, the number of constraints to be imposed on the optimization procedure underlying the learning would be very large in order to ensure a proper solution. Therefore, the number of examples needed to ensure global learning is expected to grow exponentially. It is known that networks with too many degrees of freedom (i.e., too many weights) fail to generalize, presenting several undesirable effects, such as overtraining and interference (see Boers et al., 1993). The introduction of local receptive fields reduces drastically the number of possible internal representations, allowing a relatively small number of examples to set the synapses to the proper values associated with a single internal representation.

This interpretation is consistent with our results in the NORFP, where the probability of full generalization for random target functions is much higher with the selection sampling than with a purely random sampling, even with a very small fraction of examples.

The usage of local receptive fields presents the problem of computability, that is, given a target function, we are not sure a priori if a not fully connected network like the NORFP can solve it. This is a general problem that involves not only connectivity but also the minimum number of hidden layers and the minimum number of neurons in each layer. However, specially designed networks can be constructed for particular target functions with just the prior knowledge of some global symmetries of the functions, as in the cases of addition, bit shifting, and parity described here (see also Haykin, 1994). The alternative could be a combination of the selection procedure described here with some pruning techniques, which allows reducing dynamically the connectivity of an initial fully connected network. We are working along these lines.

Finally, we showed how the scaling properties of the MNEFG for large N can be obtained by generalizing the analysis of the small network equations. The MNEFG appears as an interesting complexity parameter, consistent with the intuitive notion that the more complex the problem is, the harder it is to learn. Our results illustrate how the interplay between function and architecture complexities can be studied through this parameter. In the case of depth 2 networks, we have shown that for relatively simple functions like addition and bit shifting (though with local receptive fields), the MNEFG scales polynomially while for the more complex (parity), it scales exponentially. It is worth noting that this is the simplest depth 2 architecture that solves the parity, and hence the MNEFG, reflecting the intrinsic

complexity of this function. This is an interesting result since the parity with this architecture is one of the most used structures for testing learning algorithms. We also showed how this difficulty can be overcome by increasing the depth of the network. Once again, by using local receptive fields, the MNEFG scales polynomially with N. This is consistent with the numerical results, which show that not only with selective sampling, but even with pure random sampling, full generalization is obtained with a fraction of the examples.

Acknowledgments

This work was partially supported by the following agencies: CONICET (Argentina), CONICOR (Córdoba, Argentina), and Secretaría de Ciencia y Técnica de la Universidad Nacional de Córdoba (Argentina).

References

Baum, E. B. (1991). Neural net algorithms that learn in polynomial time from examples and queries. *IEEE Transactions on Neural Networks, 2*(1), 5.

Baum, E. B., & Haussler, D. (1989). What net size gives valid generalization? *Neural Computation, 1*(1), 151.

Boers, E. J. W., Kuiper, H., Happel, B. L. M., & Sprinkhuizen-Kuyper, I. G. (1993). Designing modular artificial neural networks. In H. A. Wijshoff (Ed.), *Proceedings of Computing Science in the Netherlands* (*CSN'93*) (p. 87). Amsterdam: SION, Stichting Matematisch Centrum.

Cannas, S. A. (1995). Arithmetic perceptrons. *Neural Computation, 7*(1), 173.

Cohn, D. (1996). Neural network exploration using optimal experiment design. *Neural Networks, 9*(6), 1071.

Cohn, D., Atlas, L., & Ladner, R. (1994). Improving generalization with active learning. *Machine Learning, 15*(2), 201.

Copelli, M., & Caticha, N. (1995). On-line learning in the committee machine. *J. Phys. A: Math. Gen., 28*, 1615.

Franco, L., & Cannas, S. A. (1998). Solving arithmetic problems using feedforward neural networks. *Neurocomputing, 18*, 61–79.

Hancock, T. R., Golea, M., & Marchand, M. (1994). Learning nonoverlapping perceptron networks from examples and membership queries. *Machine Learning, 16*, 161.

Haykin, S. (1994). *Neural networks: A comprehensive foundation*. New York: McMillan.

Hertz, J., Krogh, A., & Palmer, R. (1991). *Introduction to the theory of neural computation*. Reading, MA: Addison-Wesley and Santa Fe Institute.

Jung, G., & Opper, M. (1996). Selection of examples for a linear classifier. *J. Phys. A: Math. Gen., 29*(7), 1367.

Kinzel, W., & Ruján, P. (1990). Improving a network generalization ability by selecting examples. *Europhysics Letters, 13*(5), 473.

Minsky, M. L., & Papert, S. A. (1969). *Perceptrons*. Cambridge, MA: MIT Press.

Plutowski, M., & White, H. (1993). Selecting concise training sets from clean data. *IEEE Transactions on Neural Networks, 4*(2), 305.

Priel, A., Blatt, M., Grossman, T., Domany, E., & Kanter, I. (1994). Computational capabilities of restricted two layered perceptrons. *Physical Review E, 50*, 577.

Rumelhart, D. E., & McClelland, J. L. (1986). *Parallel distributed processing* (Vol. 1). Cambridge, MA: MIT Press.

Tessauro, G., & Janssens, R. (1988). *Scaling relationships in back-propagation learning: Dependence on predicate order* (Tech. Rep. No. CCSR-88-1). Urbana-Champaign, IL: Center for Complex System Research.

Received September 10, 1998; accepted September 26, 1999.