Biol. Cybern. 61, 125–128 (1989)

Biological Cybernetics

On the Ability of Neural Networks to Perform Generalization by Induction

V. V. Anshelevich, B. R. Amirikian, A. V. Lukashin, and M. D. Frank-Kamenetskii
Institute of Molecular Genetics, USSR Academy of Sciences, SU-123182 Moscow, USSR

Abstract. The ability of neural networks to perform generalization by induction is the ability to learn an algorithm without the benefit of complete information about it. We consider the properties of networks and algorithms that determine the efficiency of generalization. These properties are described in quantitative terms. The most effective generalization is shown to be achieved by networks with the least admissible capacity. General conclusions are illustrated by computer simulations for a three-layered neural network. We draw a quantitative comparison between the general equations and specific results reported here and elsewhere.

Introduction

For any well-defined algorithm of information processing it is possible to construct a network of neuron-like elements that will accordingly perform the input signals into output ones (McCulloch and Pitts 1943). In constructing and training such a network one should use the complete information about the algorithm that the network is to perform. Now it is sufficiently clear that potentialities of neural networks for solving deductive problems are universal (Kohonen 1977; Hinton and Anderson 1981; Rumelhart and McClelland 1986).

The ability of neural network to perform *generalization by induction* is the ability to learn an assigned algorithm without the benefit of complete information about it. Suppose that a certain part of all possible input signals has been selected and the neural network has in some way been trained to give the correct (in terms of the assigned algorithm) responses to them (output signals). After this one checks the network's reaction to all the other input signals. If all or almost all the responses are correct it means that a complete generalization has taken place; if the number of correct responses is a random value there is no generalization. Intermediate situations correspond to different degrees of generalization.

This approach dates back to the pioneering works on perceptron published in the sixties (Rosenblatt 1962; Minsky and Papert 1969), where it was demonstrated, using specific examples, that little or no generalization takes place unless a priori information about the algorithm is introduced into the neural network. At present we know of different network constructions and training procedures (Hinton 1987; Patarnello and Carnevali 1987; Carnevali and Patarnello 1987; Anderson 1988; Zipser and Andersen 1988; Scalettar and Zee 1988; Lehky and Sejnowskii 1988) that are associated with different degrees of generalization efficiency, including fairly high ones. The "record" seems to have been set in (Patarnello and Carnevali 1987), where complete generalization was achieved after the network had been trained to give the correct responses to 0.3% of the possible input signals.

The goal of the present paper is to find out which properties of the networks and of the algorithms that they are trained determine the efficiency of generalization. We present a quantitative characterization of these properties in terms of information theory and illustrate general conclusions by computer simulations.

The Capacity of a Network

Let a neural network have a fixed number n_{in} of input neurons and n_{out} output neurons each of which can be in one of two states: 0 or 1. Besides, there are n_{hid} hidden neurons. The input signal is a binary vector of n_{in} components that is fed to the input neurons. The work of the neural network produces a response at the output neurons; the response is a binary vector of n_{out}

components. The signal transformation algorithm is defined by the set of variable parameters, which usually include the threshold characteristics of the neurons and the characteristics of their interaction. In the training procedure these parameters are changed, leading to a change of the information processing algorithm. *The capacity of a network* is characterized by the number of different algorithms, N, that can be performed by this network through changing the values of the variable parameters. N grows with increasing number of hidden neurons n_{hid}.

Suppose an algorithm that a network must learn can indeed be performed by it. Then the amount (bits) of information one has to feed to the network in order that this particular algorithm might be performed is equal to $-\log_2 P$, where P is the probability of the untrained network giving the correct responses to all the input signals. On the other hand, as a result of learning the correct responses to m input signals the network gets $mn_{\text{out}}h$ bits of information, where $h=-\log_2 p$, p being the probability of the correct response in one output neuron, hereinafter assumed to be 0.5 (for the sake of simplicity). Hence, for complete generalization to take place one has to train the network to give correct responses to m^* input signals, this value being defined by the equation $m^* n_{\text{out}} = -\log_2 P$. Assuming all the algorithms that can be performed by the network to be equally probable, i.e. assuming $P=1/N$, we have

$$m^* = n_{\text{out}}^{-1} \log_2 N . \tag{1}$$

The values m^* or m^*/M, where $M=2^{n_{\text{in}}}$ is the total number of input signals, characterize the efficiency of generalization. The smaller these values the higher the generalization efficiency. Of course, in the general case (1) gives only a rough estimate of the generalization efficiency, since it is based on a number of strong assumptions. However, in our view this shortcoming is compensated by clarity and independence of (1) of the structure details, as well as the rules of performance and training procedure of the neural network under consideration. Furthermore, as we are going to demonstrate below, (1) gives a satisfactory quantitative description of the generalization efficiency for specific examples.

Of special importance to us is the inference from the above consideration in terms of information theory to the effect that the efficiency of generalization deteriorates with increasing capacity of the neural network. The least number of hidden neurons which still enables the network to perform the assigned algorithm is the optimum situation in terms of the best generalization effect. The maximum attainable generalization efficiency is determined, for a given algorithm, by the capacity of the minimal network.

Simulations and Discussion

Now consider, by way an illustration, the generalization ability of a three-layered feed forward neural network. Let binary vectors $\{x_i\}$, $\{y_j\}$, and $\{r_k\}$ where $i=1,\ldots,n_{\text{in}}$, $j=1,\ldots,n_{\text{out}}$, $k=1,\ldots,n_{\text{hid}}$, describe the states of the input, output and hidden neurons respectively. When a signal is fed to the input neurons, the states of the hidden and output neurons are formed according to the following rule:

$$r_k = \Theta\left(\sum_i a_{ki} x_i - t_k\right), \qquad y_j = \Theta\left(\sum_k b_{jk} r_k\right), \tag{2}$$

$$\Theta(z) = \begin{cases} 1, & z>0 \\ 0, & z\leqq 0. \end{cases} \tag{3}$$

The parameters a_{ki} and b_{jk} describe inter-neuron connections and assume the following values: $a_{ki}=-1,0,1$; $b_{jk}=0,1$. The threshold $t_k=s_k-1$, where s_k is the number of positive connections ($a_{ki}=1$) between the input neurons and the k-th hidden neuron. For a large enough number of hidden neurons, namely for $n_{\text{hid}}\geqq 2^{n_{\text{in}}}$, this network is universal, i.e. it can perform any algorithm.

The neural network was trained in the following way. A random choice of m examples was made out of the total set of $M=2^{n_{\text{in}}}$ input signals. Then the "annealing" procedure (Kirkpatrick et al. 1983) was used for minimizing the error function for this collection of m input signals. The error function was defined in the following way:

$$E_m = (mn_{\text{out}})^{-1} \sum_{j=1}^{n_{\text{out}}} \sum_{l=1}^{m} |y_j^{(l)} - \bar{y}_j^{(l)}|, \tag{4}$$

where $y_j^{(l)}$ is the response obtained and $\bar{y}_j^{(l)}$ is the correct response to the l-th input signal. The values of a_{ki}, t_k, and b_{jk} which make the function (4) equal to zero were found by "annealing" and fixed, then error function E_M was calculated for the *total set* of M input signals. Then this procedure was repeated for another realization of m examples, and so on. The sought-for characteristic is the total mean error $E=\langle E_M \rangle$ for random realization with a fixed number m.

For the network described above one can compare the values m/M for which E is close to zero, i.e. the values already sufficient for a complete generalization, with the values m^*/M yielded by (1). To do this one has to know the capacity of the network N. We estimated it as $N=N_s/n_{\text{hid}}!$, where N_s is the number of possible configurations of the network, and the denominator $n_{\text{hid}}!$ is associated with possible permutations of hidden neurons. Then it follows from (1) that

$$m^*/M = n_{\text{hid}}(n_{\text{in}} \log_2 3 + n_{\text{out}} - \log_2(n_{\text{hid}}/e))/(n_{\text{out}} M). \tag{5}$$

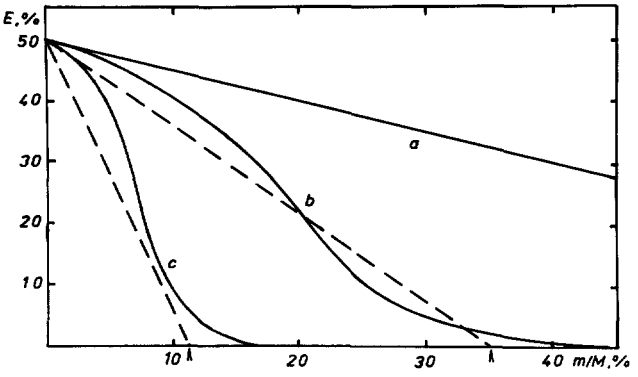

Fig. 1. The results of calculations of the total mean error E (see text) versus the proportion of input signals used in training of the network. Curves: *a* $n_{in}=4$, $n_{hid}=4$; *b* $n_{in}=6$, $n_{hid}=6$; *c* $n_{in}=8$, $n_{hid}=8$. Arrows mark the values m^*/M following from (5): the right arrow corresponds to parameters of curve *b*, the left one corresponds of curve *c*. The dashed lines are shown for the sake of clearity

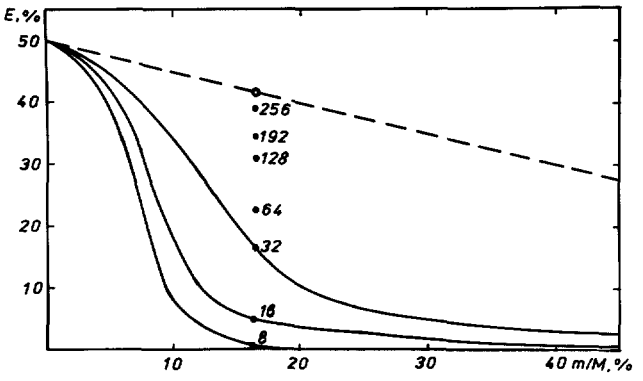

Fig. 2. The dependence of the efficiency of generalization on the capacity of the network. The error E is shown for different numbers of hidden neurons n_{hid} with a fixed number of input neurons $n_{in}=8$. The figures indicate the values of n_{hid}. The entire dependence of E on m/M is shown for $n_{hid}=8$, 16, and 32. For other values of n_{hid} we only show the E values for $m/M=16.4\%$. The dashed line illustrates the E function in the absence of all generalization

As an example, let us consider the addition of two binary vectors as the algorithm to be learned. Let $n_{in}=2n_{out}$. Then the network must learn to transform any input signal $\{x_i\}$ $(i=1,\ldots,n_{in})$ into the output signal $\{y_j\}$ $(j=1,\ldots,n_{out})$ according to the rule $y_j=x_j+x_{j+n_{out}}$, the addition being defined by the table: $0+0=1+1=0$, $0+1=1+0=1$.

Figure 1 shows the results of calculations, for the case of the above algorithm, of the total mean error E depending on the number of examples m used in training of the network, for different values of n_{in}.

In Fig. 1 the values m^*/M, which are obtained from (5) for $n_{in}=6$ and 8, are marked by arrows. One can see that (5) gives an almost complete quantitative agreement with the results of the computer experiments. A similar situation arises for $n_{in}=4$, 10, 12, 14 (not shown). For instance, for $n_{in}=14$ the complete generalization is attained, according to (5), if the network is trained to give correct responses to 0.33% input signals ($m^*=54$, $M=2^{14}$). The value produced by our computer simulations for this case lies within the 0.35–0.40% range.

In similar manner one can use (1) to explain the results obtained in (Patarnello and Carnevali 1987). It is particularly important because in that case the construction of the network and the properties of the elements (neurons) were different from ours. The complete number of network configurations in (Patarnello and Carnevali 1987)

$$N_s=((n_{in}+n_{hid}+n_{out}-1)!/(n_{in}-1)!)^2. \quad (6)$$

Disregarding the possible degeneracy, let N_s be the estimate for N. Then for the parameters of (Patarnello and Carnevali 1987) $n_{in}=16$, $n_{out}=8$, $n_{hid}+n_{out}=160$ it follows from (1), (6) that $m^*=254$, which agrees to within 10% with the experimental value $m^*=224$ obtained in (Patarnello and Carnevali 1987).

In the calculations presented in Fig. 1 the number of hidden neurons was equal to their minimal number for which three-layered network described above can perform the algorithm of addition of two binary vectors ($n_{hid}=n_{in}$). According to (1) the generalization efficiency must decrease with increasing number of hidden neurons, i.e. with greater network capacity. We checked this for our model. Figure 2 shows the results of calculations of the total mean error E at the fixed value of $n_{in}=8$ for different numbers of hidden neurons. One can see the monotonic decrease of the generalization efficiency with increasing n_{hid}; at $n_{hid}=256$ the E value is close to the error value at which there is no generalization at all, just memorization is achieved. Hence, there exists an optimum number of hidden neurons ensuring the maximum generalization efficiency, for a given algorithm, (cf. p. 507 in Patarnello and Carnevali 1987).

Thus, our computer simulations agree very well with (1), which estimates the network capacity via the number of possible configurations.

Our experience shows that this is the case also for other simple algorithms. The question whether the same is true for complex algorithms remains to be answered. However, main qualitative conclusion, that the generalization efficiency drops monotonously with increasing of the network capacity, must not depend on the algorithm complexity. We therefore believe that the major practical recommendation, which follows from our data, is of general validity. Namely, the maximum generalization efficiency is achieved for the network with minimum number of the hidden neurons that is sufficient to execute the algorithm under study.

References

Anderson A (1988) Learning from a computer cat. Nature 331:657–659

Carnevali P, Patarnello S (1987) Exhaustive thermodynamical analysis of Boolean learning networks. Europhys Lett 4:1199–1204

Hinton GE (1987) Learning translation invariant recognition in a massively parallel network. In: Proceedings of the Conference on Parallel Architectures and Languages in Europe. Springer, Berlin Heidelberg New York

Hinton GE, Anderson JA (1981) Parallel models of associative memory. Erlbaum, Hillsdale

Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680

Kohonen T (1977) Associative memory: a system theoretic approach. Springer, Berlin Heidelberg New York

Lehky SR, Sejnowskii T (1988) Network model of shape-from-shadding: neural function arises from both receptive and projective fields. Nature 333:452–454

McCulloch WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys 5:115–133

Minsky M, Papert S (1969) Perceptrons. MIT Press, Cambridge

Patarnello S, Carnevali P (1987) Learning networks of neurons with Boolean logic. Europhys Lett 4:503–508

Rosenblatt F (1962) Principles of neurodynamics. Spartan Books, Washington

Rumelhart DE, McClelland JL (1986) Parallel distributed processing: Explorations in the microstructure of cognition, vols 1, 2. MIT Press, Cambridge

Scalettar R, Zee A (1988) Perception of left and right by a feed forward net. Biol Cybern 58:193–201

Zipser D, Andersen RA (1988) A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. Nature 331:679–684

Received: July 14, 1988
Accepted in revised form: February 2, 1989

Prof. M. D. Frank-Kamenetskii
Institute of Molecular Genetics
Academy of Sciences of USSR
SU-123182 Moscow
USSR