

COMPLEXITY IN NEURAL SYSTEMS

Yaser Abu-Mostafa

Why do neurons have so many inputs? Although we do not have a complete understanding of the operation of individual neurons and of large neural systems, we know enough to see why the large number of neuron inputs matches the basic function of the neurons and the system. Two of the factors contributing to this match are what we call the **analog factor** and the **entropy factor**. We start by giving a brief description of these factors; then we discuss each in detail.

1. **Analog factor:** The analog mode of computation makes the computation power of an individual neuron proportional to the *square* of the number of inputs K , whereas the neuron's size is only linear in K . This result contrasts with traditional logic gates, such as AND gates, in which both computation power and size are linear in K , and hence would be equally well implemented by a tree of two-input gates of the same type. To simulate the function of a neuron with K inputs, we need the order of K^2 two-input neurons. Therefore, the saving in overall system size when we use the K -input neurons is proportional to K , and is several orders of magnitude for the known values of K in the brain.
2. **Entropy factor:** The typical input for the neural system comes from a natural environment that has a certain degree of disorder, or entropy. **Entropy** is a quantitative measure of the disorder or randomness of an

environment. An important part of the function of the neural system is to be able to learn from "training" samples drawn from the environment. Under what conditions is learning possible? If we assume that the learning mechanism is local, as in the case of Hebbian learning, where the strength of a synapse is incremented or decremented according to the states of the two neurons it connects, we can show that a relation holds between the entropy of the environment and the number of neuron inputs. The relation forces the number of neuron inputs to be at least equal to the entropy of the environment.

ANALOG FACTOR

A neuron, like any other logic device, makes a decision based on the values of its inputs. However, the decision-making mechanism in the case of neurons is analog; that is, it involves the processing of continuous-valued signals rather than of discrete-valued signals. For example, the function of certain neurons can be modeled as a **threshold rule**: The neuron will fire (will have output +1) if the weighted sum of its inputs exceeds an internal threshold; otherwise, it will not fire (will have output -1). Thus,

$$u_i = \begin{cases} +1, & \text{if } \sum_{j=1}^K w_{ij} u_j \geq t_i \\ -1, & \text{if } \sum_{j=1}^K w_{ij} u_j < t_i \end{cases} \quad (\text{D.1})$$

where u_i is the output of neuron i , $\{u_j\}$ are the inputs to this neuron (and also are the outputs of other neurons), $\{w_{ij}\}$ are the weights of the synaptic connections, and t_i is the internal threshold. Although, in this equation, the inputs u_1, u_2, \dots, u_K and the output u_i are all discrete (binary), output depends on the input through the analog parameters $\{w_{ij}\}$ and t_i . The function of most neurons is more sophisticated than is this simple threshold rule; some neurons accept analog values for their inputs and generate analog outputs.

Why does the analog processing mode make the large number of inputs to the neuron vital? The techniques of circuit complexity provide a direct answer. Consider the neuron described by Equation D.1 for u_i . Suppose we replace this K -input neuron by a network of two-input neurons that does the same job. The idea of this hypothetical replacement is to see whether there is an inherent advantage to having neurons with a large number of inputs K , or whether we can do equally well with more neurons, each of which has fewer inputs. For example, if we considered K -input AND gates instead of K -input neurons, the replacement network would be a tree of two-input AND gates (see Figure D.1). Let us compare the sizes of the single K -input gate and of the tree. The size of the K -input gate is approximately proportional to K (it may vary from one physical implementation to another). The size of the tree is proportional to the number of two-input gates in the tree, which is approximately K . Hence, there is

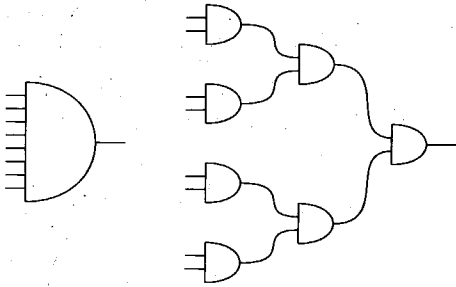


FIGURE D.1 A tree of two-input AND gates (right) replacing a single eight-input AND gate (left) and performing the same function.

no essential gain or loss in the overall system size if we use only two-input gates. Does the situation change when we consider neurons instead of AND gates?

It does. Here also, the size of the K -input neuron is approximately proportional to K . However, the neuron can be replaced by only a network of two-input neurons the size of which is proportional to *at least* K^2 . Let us first see why the size of the K -input neuron is proportional to K . The synapses merge toward the neuron (see Figure 7.1, p. 102) in a treelike fashion (not necessarily a binary tree). As they merge, their signals ($w_{ij}u_j$) are added, forming partial sums that accumulate to create the total signal $\sum_{j=1}^K w_{ij}u_j$. The neuron then compares the signal to the internal threshold t_i and determines the value of u_i . The overall size of the neuron including this merging tree is dominated by the size of the tree, which is proportional to K . Hence, the size of the K -input neuron is proportional to K , as we asserted. So why is it not possible to replace the K -input neuron by a tree of two-input neurons in exactly the same way as we did for the AND gate?

Because the neuron simulates a general threshold function of K variables, the network of two-input neurons should be able to simulate any threshold function of K variables. Any such network, however, must have at least the order of K^2 two-input neurons, a consequence of the circuit complexity of threshold functions. Suppose we have a network with M two-input neurons that can simulate any threshold function of K inputs. There are only 16 ways to program each neuron, because there are only 16 switching functions of two variables. Hence, there are at most 16^M different ways to program the network to simulate a given overall function. This number must be at least large enough to accommodate the number of different K -input threshold functions, which is at least $2^{K^2/2}$ [Muroga et al., 1966]. Hence,

$$16^M \geq 2^{K^2/2}$$

which forces M to be at least the order of K^2 .

The argument does not guarantee that there is a network with K^2 two-input neurons that can replace the K -input neuron. It shows only that there is no smaller network that will do it, which is exactly what we were trying to prove. The situation becomes even worse when logic more sophisticated than threshold is involved. The fancier the logic, the more it pays to have plenty of inputs to the neuron. The ability of the neurons to handle analog signals makes it worthwhile

to save the analog nature of the data and then to make a global decision. With a network of two-input neurons, the intermediate discrete decisions waste most of the information carried by the intermediate analog signals, and a bigger network is needed to make up for this lost analog information by providing an abundance of intermediate discrete decisions. For example, the intermediate analog signals may be soft decisions (analog values between 0 and 1, instead of just 0 or 1) carrying not only the decision (0 or 1, depending on which is closer), but also information about how reliable the decision is (how close). We need several bits to encode an approximate value of this reliability if we use discrete signals.

ENTROPY FACTOR

The ability of neural systems to learn spontaneously a desired function from training samples is these systems' most important feature. Clearly, a given neural system cannot learn any old function; there must be restrictions on which systems can learn which functions. We present a theorem describing such a restriction. The theorem imposes a lower bound on the number of neuron inputs, in terms of the entropy of the environment. For the theorem to hold, the learning mechanism is assumed to be local; when a training sample is loaded into the system, each neuron has access to only those bits that are carried by it and by the neurons to which it is directly connected. This strong assumption excludes the sophisticated learning mechanisms used in neural-network models, but it may be more plausible from a biological point of view.

The environment in our model produces patterns represented by N bits $\mathbf{x} = x_1 \cdots x_N$. Only h different patterns can be generated by a given environment, where $h < 2^N$ (the entropy is essentially $\log_2 h$). We do not assume any knowledge about which patterns the environment is likely to generate—we know only that there are h such patterns. In the learning process, a huge number of sample patterns are generated at random from the environment and are sent to the system, 1 bit per neuron. The system uses this information to set its internal parameters and gradually to tune itself to this particular environment. Because of the system architecture, each neuron knows only its own bit and (at best) the bits of the neurons to which it is directly connected by a synapse. Hence, the learning rules are local: A neuron does not have the benefit of the entire global pattern that is being learned.

After the learning process has taken place, each neuron is ready to perform a function defined by what it has learned. The collective interaction of the functions of the neurons is what defines the overall function of the network. The theorem reported here is that (roughly speaking), if the number of neuron inputs is less than the entropy, the system cannot learn about the environment. The idea used to prove the theorem is to show that, if the number of neuron inputs is small, then the final function of each neuron is independent of the environment. Hence, we can conclude that, in this situation, the overall system has accumulated no information about the environment. Let us formalize these ideas.

A neural system is an undirected graph (the vertices are the neurons and the edges are the synapses). We label the neurons $1, \dots, N$ and define $K_n \subseteq \{1, \dots, N\}$ to be the set of neurons connected by a synapse to neuron n , together with neuron n itself. An environment is a subset $e \subseteq \{0, 1\}^N$ (each $\mathbf{x} \in e$ is a sample from the environment). During learning, x_1, \dots, x_N (the bits of \mathbf{x}) are loaded into the neurons $1, \dots, N$, respectively. Now we consider an arbitrary neuron n and relabel everything to make K_n become $\{1, \dots, K\}$. Thus, the neuron sees the first K coordinates of each \mathbf{x} .

Our result is asymptotic in N , so we specify K as a function of N ; $K = \alpha N$, where $\alpha = \alpha(N)$ satisfies $\lim_{N \rightarrow \infty} \alpha(N) = \alpha_o$, for $0 < \alpha_o < 1$. The result also is statistical, so we consider the *ensemble* of environments \mathcal{E}

$$\mathcal{E} = \mathcal{E}(N) = \{e \subseteq \{0, 1\}^N : |e| = h\}$$

where $h = 2^{\beta N}$, and $\beta = \beta(N)$ satisfies $\lim_{N \rightarrow \infty} \beta(N) = \beta_o$, for $0 < \beta_o < 1$. The probability distribution on \mathcal{E} is uniform; any environment $e \in \mathcal{E}$ is as likely to occur as is any other.

The neuron sees only the first K coordinates of each \mathbf{x} generated by the environment e . For each e , we define the function $n: \{0, 1\}^K \rightarrow \{0, 1, 2, \dots\}$, where

$$n(a_1 \cdots a_K) = |\{\mathbf{x} \in e : x_k = a_k \text{ for } k = 1, \dots, K\}|$$

and the normalized version

$$\nu(a_1 \cdots a_K) = \frac{n(a_1 \cdots a_K)}{h}$$

The function ν describes the relative frequency of occurrence for each of the 2^K binary vectors $x_1 \cdots x_K$ as $\mathbf{x} = x_1 \cdots x_N$ runs through all h vectors in e . In other words, ν specifies the projection of e as seen by the neuron. Clearly, $\nu(\mathbf{a}) \geq 0$ for all $\mathbf{a} \in \{0, 1\}^K$ and $\sum_{\mathbf{a} \in \{0, 1\}^K} \nu(\mathbf{a}) = 1$.

Corresponding to two environments e_1 and e_2 , we will have two functions ν_1 and ν_2 . If ν_1 is not distinguishable from ν_2 , the neuron cannot tell the difference between e_1 and e_2 . The distinguishability between ν_1 and ν_2 can be measured by

$$d(\nu_1, \nu_2) = \frac{1}{2} \sum_{\mathbf{a} \in \{0, 1\}^K} |\nu_1(\mathbf{a}) - \nu_2(\mathbf{a})|$$

The range of $d(\nu_1, \nu_2)$ is $0 \leq d(\nu_1, \nu_2) \leq 1$, where 0 corresponds to complete indistinguishability and 1 corresponds to maximum distinguishability. Let e_1 and e_2 be independently selected environments from \mathcal{E} according to the uniform probability distribution. Now $d(\nu_1, \nu_2)$ is a random variable, and we are interested in the expected value $E(d(\nu_1, \nu_2))$. The case where $E(d(\nu_1, \nu_2)) = 0$ corresponds to the neuron getting no information about the environment; the case where $E(d(\nu_1, \nu_2)) = 1$ corresponds to the neuron getting maximum information. The following theorem predicts, in the limit, one of these extremes, depending on how the number of neuron inputs (represented by α_o) compares to the entropy (represented by β_o).

Theorem

1. If $\alpha_o > \beta_o$, then $\lim_{N \rightarrow \infty} E(d(\nu_1, \nu_2)) = 1$
2. If $\alpha_o < \beta_o$, then $\lim_{N \rightarrow \infty} E(d(\nu_1, \nu_2)) = 0$

A formal proof of the theorem is given in previous publications [Abu-Mostafa, 1988a; Abu-Mostafa, 1988b]. Here, we provide a sketch of that proof. Suppose $h = 2^{K+10}$ (corresponding to part 2 of the theorem). For most environments $e \in \mathcal{E}$, the first K bits of $\mathbf{x} \in e$ go through all 2^K possible values approximately 2^{10} times each as \mathbf{x} goes through all h possible values once. Therefore, the patterns seen by the neuron are drawn from the fixed ensemble of all binary vectors of length K with essentially uniform probability distribution; that is, ν is the same for most environments. This observation means that, statistically, the neuron will end up performing the same function regardless of the environment in question.

What about the opposite case, where $h = 2^{K-10}$ (corresponding to part 1 of the theorem)? Now, with only 2^{K-10} patterns available from the environment, the first K bits of \mathbf{x} can assume at most 2^{K-10} values out of the possible 2^K values a binary vector of length K can assume in principle. Furthermore, which values can be assumed depends on the particular environment in question; that is, ν depends on the environment. Therefore, although the neuron still does not have the global picture, the information it has provides some information about the environment.

In summary, we have contrasted two situations. In the first situation, the neuron sees too few bits and cannot observe any pattern in these bits that identifies the environment. In the second situation, the neuron sees enough bits to form up such a pattern. The critical value of the number of bits that differentiates between the two situations is equal to the entropy of the environment.

REFERENCES

- Abu-Mostafa, Y. Connectivity versus entropy. In Anderson, D. (ed), *Neural Information Processing Systems*. New York: American Institute of Physics, 1988a, p. 1.
- Abu-Mostafa, Y. Lower bound for connectivity in local-learning neural networks. *Journal of Complexity*, 4:246, 1988b.
- Muroga, S. and Toda, I. Lower bound of the number of threshold functions. *IEEE Transactions on Electronic Computers*, EC-15:805, 1966.