1
Neural Networks: A Control Approach

Introduction

A *neural network* is a network of subunits, called "formal neurons," processing input signals to output signals, which are coupled through "synapses." The synapses are the nodes of this particular kind of network, the "strength" of which, called the *synaptic weight*, codes the "knowledge" of the network and controls the processing of the signals.

Let us be clear at the outset that the resemblance of a formal neuron to an animal-brain neuron is not well established, but that is not essential at this stage of abstraction. However, this terminology can be justified to some extent, and it is by now widely accepted, as discussed later. Chapter 8 develops this issue.

Also, there is always a combination of two basic motivations for dealing with neural networks - one attempting to model actual biological nervous systems, the other being content with implementation of neural-like systems on computers. Every model lies between these two requirements - the first constraining the modeling, the second allowing more freedom in the choice of a particular representation.

There are so many different versions of neural networks that it is difficult to find a common framework to unify all of them at a rather concrete level. But one can regard neural networks as dynamical systems (discrete or continuous), the states of which are the signals, and the controls of which are the synaptic weights, which regulate the flux of transmitters from one neuron to another. They yield what are also known as *adaptive systems*, controlled by *synaptic matrices*.

We investigate in this chapter the *supervised learning* of a finite set of patterns, called the *training set*, each pattern being a pair of input-output signals. A *learning process* amounts to matching the given input

signals of the patterns of this training set with the associated output signals. It thus involves a comparison with desired answers, done by a "supervisor." Hence, the learning problem amounts to finding a synaptic matrix that can "learn" the patterns of a given training set. Any algorithm yielding such a synaptic matrix will then be regarded as a *learning rule*. This seems to exclude neural networks from reasonable models of basic neural functions. However, the problems under investigation here are as follows:

1. Establish the existence of exact (or approximate) synaptic matrices that have learned a given set of patterns,
2. Find algorithms, regarded as learning rules, that will provide a sequence of synaptic matrices converging to a solution of the learning problem.

I propose in this first chapter a short presentation of the learning processes of neural networks in the framework of dynamical systems controlled by synaptic matrices. Section 1.2 explains how neural networks operate to solve pattern-classification problems and, in particular, to extrapolate time series in forecasting problems. These problems are called supervised learning problems, because the patterns to be taught are provided by a supervisor, so to speak.

Biological Comments. Although the results to be derived here will be "formal" and will not necessarily be associated with any biological implementation, it may be useful to provide a crude description of the nature of the processing carried out by some biological neurons (Figure 1.1).

It should be first pointed out that many kinds of neuronal cells evolved during phylogenesis, each of them selected to provide adequate technical solutions to biological or environmental problems. Most neurons in the central nervous system in higher animals can be regarded as *impulse oscillators*. They produce trains or volleys of neural impulses whose average frequency will depend on the input excitation. The processing carried out by neuron ensues from biophysical and biochemical phenomena in the membrane of the neuron, wherein functions the machinery that controls the interaction of the cell and its environment.

Models of neural dynamics that attempt to describe the triggering phenomena, the transmission of the impulses, and their biochemical control have been extensively studied since the Hodgkin-Huxley model was first proposed. The central features are the following: There is a static

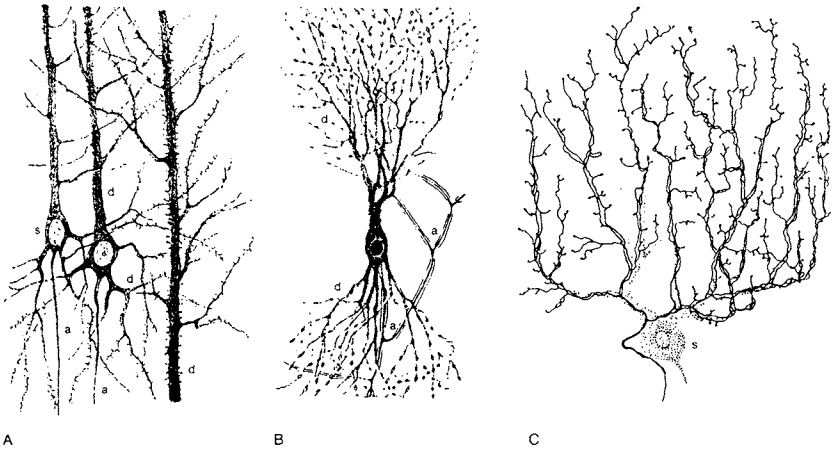

Fig. 1.1. Neurons. Pyramidal and Purkinje neurons

electrical-potential difference of about 70 mV between the inside of the neuron (negatively charged) and its environment; it is maintained by diffusion of ions through the membrane. The synaptic receptors are able to control the ionic conductance of the membrane through a highly sophisticated biochemical and ionical mechanism (Figure 1.2).

When neurotransmitters (excitation signals) arrive at a receptor, the excitatory synapses tend to produce a depolarization of the membrane, and the inhibitory synapses tend to produce a hyperpolarization. When the sum (i.e., algebraic sum) of the depolarizations exceeds a given threshhold, then the membrane's permeability to ions is increased, and the membrane becomes electrically active, sending an output impulse of about 100 mV amplitude during a period of 0.5-2 ms. After each impulse, there is a short refractory period during which the membrane recovers so as to be ready for the next impulse (Figure 1.3).

Because in our framework we are interested in the collective processing by neurons, we shall be content with only a crude analytical description of the processing role of the neuron. We retain only the following features: The impulse frequency oscillates between bounds determined by physical and chemical factors (oscillating between 0 and 500 Hz). Between those bounds, the postsynaptic *average oscillatory frequency* is assumed to be a monotonic function of the net algebraic sum of the presynaptic average frequencies of the inputs afferent to the neurons (inhibitory excitations being regarded as negative excitations). Even though, for simplicity, the integration of the presynaptic inputs is often

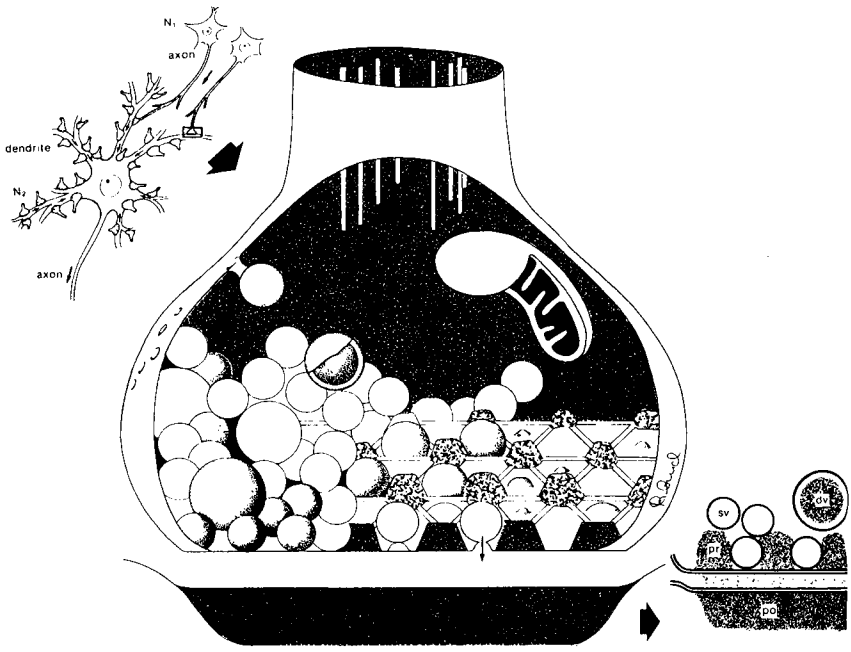

Fig. 1.2 Synapses. A larger view of a synaptic terminal, showing the vesical grid, the vesicles incorporated in the cell membrane, a single mitochondrion and the cleft, but not the postsynaptic detail.

chosen to be linear, biological neurons are somewhat "leaky integrators" of presynaptic inputs, certainly nonlinear.

1.1 Neural Networks: A General Presentation

A way to encompass most of the neural networks - also called *parallel distributed processes* (PDP) - studied in the literature is to regard them as dynamical systems controlled by synaptic matrices.

1.1.1 Formal Neurons

We begin with a set of n "formal neurons" (or abstract neurons, processing units, etc.) labeled by $j = 1, \ldots, n$. The processing of a neural network is carried out by these formal neurons. The formal neuron's job is simply to receive afferent signals from the other formal (presynaptic) neurons (or the input signals) and to provide (process, compute, etc.) an output that is sent to the other (postsynaptic) neurons (or to

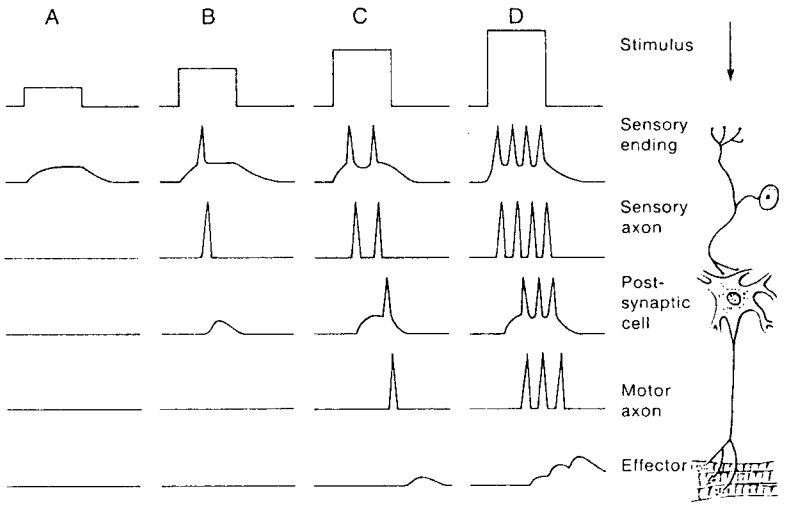

Fig. 1.3 Propagation of the Nervous Influx. A stimulus is applied to a sensory nerve terminal with increasing intensityfrom A to B.

the output of the network). A neural network is *parallel* in that many (if not all) neurons carry out their processing at the same time. This processing is said to be *synchronous* if the duration of the processing of a unit of input is the same for all formal neurons. If not, it is called *asynchronous*. Synchronous models can be regarded as discrete models. Biological neurons are asynchronous and thus require a continuous-time treatment, through differential equations or inclusions.

1.1.2 Signals: The States of the Network

Formal neurons link (directly for one-layer networks, and indirectly for multiple-layer networks) an input space X of signals to an output space Y. Hence, the *state space* of the system is the space $X \times Y$ of input-output pairs (x, y), which are called *patterns* in pattern recognition, data analysis, and classification problems. The set of input signals is often called the *retina* in the framework of image recognition, for obvious reasons. When the inputs are different from the outputs, the system is called *heteroassociative*. When $X = Y$ and when the inputs of the patterns (x, x) coincide with the outputs, we speak of *autoassociative* networks.

We have to distinguish among all possible input-output patterns a subset $\mathcal{K} \subset X \times Y$ of *patterns*, called the *training set*. For instance, the input signals may obey state constraints. The most common ones require that each state of excitation x_i lie in the interval $[-1, +1]$ (Anderson's model of "brain state in a box") or in any other interval, depending upon the nature of the signals. For neural networks dealing with Boolean functions, the inputs range over $\{0,1\}^n$, or $[0,1]^n$ in the case of fuzzy statements. In the case of autoassociative networks, where $Y = X$, the training set $\mathcal{K}$ is contained in the diagonal $\{(x,x)\}_{x \in X}$. Most often, the input and output spaces are finite-dimensional vector spaces $X := \mathbf{R}^n$ and $Y := \mathbf{R}^m$.

In biological neurons, the signal is represented by the average oscillatory frequency of the nervous impulses, or by the short-term average firing rate (or triggering frequency), or by the concentration of neurotransmitters in the synapse at a given time.

1.1.3 Synaptic Matrices: The Controls of the Network

Formal neurons are connected to one another in a neural network. The conventional wisdom among neural-network scientists is that *knowledge* is encoded in the pattern of connectivity of the network. Let N denote the set of neurons, and 2^N or $\mathcal{P}(N)$ the family of subsets of neurons, called *conjuncts* or *coalitions* (of neurons) (Figure 1.4).

The connection links a postsynaptic neuron j to conjuncts or coalitions $S \subset N$ of presynaptic neurons. Each conjunct S of neurons preprocesses (or gates[1]) the afferent signals x_i produced by the presynaptic neurons through a function

$$\varphi_S \ : \ x := (x_j)_{j=1,\ldots,n} \mapsto \varphi_S(x)$$

In most models, the conjuncts $S = \{i\}$ are reduced to individual neurons i. Then the role of the control is played by the synaptic matrix

$$W = \left(w_j^S\right)_{\substack{S \in 2^N \\ j=1,\ldots,n}}$$

the entries w_j^S of which are the synaptic weights. The modulus of the synaptic weight represents the strength, and its sign the direction, of the connection from the conjunct S to the formal neuron j, counted positively if the synapse is excitatory, and counted negatively if it is

[1]This gating process is useful to compute (or extrapolate) Boolean functions.

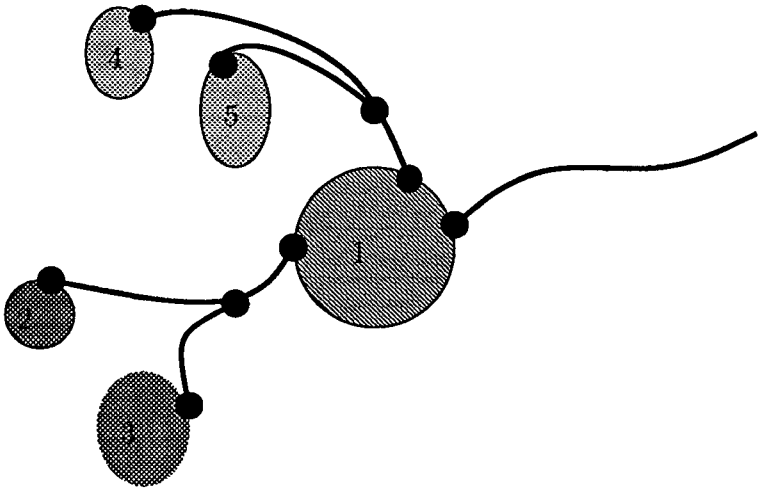

Fig. 1.4 Conjucts of Neurons. Two conjucts of neurons $\{2,3\}$ and $\{4,5\}$ provide inputs to the post-synaptic neuron 1.

inhibitory. Therefore, it is often assumed in this kind of neuromimetic model that the j^{th} neuron is excited by $\sum_{S\in 2^N} w_j^S \varphi_S(x)$.

Some models impose constraints bearing on the structure of the synaptic matrix, which may represent a division of the set of neurons in several layers, with the synaptic matrix as a "product" of elementary synaptic matrices mapping one layer to another (top-down or bottom-up processing systems). There is no need to assume that a synaptic matrix is symmetric. Actually, experimental evidence suggests asymmetric matrices, and perhaps antisymmetric matrices.

1.1.4 Propagation Rules: The Dynamics of the Network

The output of the jth neuron is a function of the neuron potential caused by the preprocessed signals sent by the presynaptic neurons, (possibly) gated by conjuncts of neurons and weighted by the synaptic weights that tune these presynaptic neuron potentials.

The evolution of the average is governed by a discrete dynamical system (for synchronous neurons),

$$y_j := f_j\left(\left\{(w_j^S), \varphi_S(x)\right\}_{S\in 2^N}\right)$$

or by a differential equation of the form

$$x_j'(t) = f_j\left(\left\{(w_j^S)(t), \varphi_S(x(t))\right\}_{S\in 2^N}\right)$$

for asynchronous neurons.

Most often, the function f_j is described in the following form:

$$f_j\left(\left\{(w_j^S), \varphi_S(x)\right\}_{S\in 2^N}\right) := g_j\left(\sum_{S\in 2^N} w_j^S \varphi_S(x)\right)$$

where g_j represents the integrator of the afferent signals, and where $w_j^S\varphi_S$ describes the signal sent to the formal neuron j when it is excited by the inputs x_i, preprocessed by the conjunct S and delivered to neuron j through the weight w_j^S. [Usually, one assumes that $w_j^S = 0$ when $j \in S$. The case when $w_j^S \neq 0$ when $S \ni j$ allows feedback (or autoexcitation) in the neural network.]

When $S = \{j\}$ and when the synaptic weights w_i^S $(S \neq \{i\})$ are zero, we obtain the *loss term* $g_i(0, \ldots, w_j^j\varphi_j(x), \ldots, 0)$, which may represent "forgetting," or at least the decaying of the signal frequency when the neuron is not excited by the other neurons.

To be realistic, delays and, more generally, the history of $x(\cdot)$ should appear in these functions φ_S.

1.2 Examples of Neural Networks

Let us cite several examples of propagation rules:

1. Associative Memories

The richest class of neural networks is obtained when there is no preprocessing by conjuncts of neurons, that is, when $\varphi_S(x) = 0$ if $|S| > 1$ and when $\varphi_{\{i\}}(x) = x_i$ for any afferent signal i (where $|S|$ denotes the number of elements of S) and when the functions g_i are affine: They are governed by the simple law

$$y_j = \sum_{i=1,\ldots,n} w_j^i x_i + c_j$$

Such networks are called *associative memories* by Kohonen. We shall answer most questions in this familiar setting. When the dimension of the output space $Y := \mathbf{R}$ is 1, we find the Widrow *adaline network* (for "adaptive linear element") introduced in 1962.

2. Associative Memories with Gates

When the functions g_i are still affine and there is preprocessing of the

inputs of an associative memory through conjuncts of neurons, we obtain the associative memories with gates:

$$y_j := \sum_{S \in 2^N} w_j^S \varphi_S(x) + c_j$$

Example: Boolean Associative Memories

Denoting by $|S|$ the number of elements of a conjunct S (a subset of neurons), an afferent signal $x \in \mathbf{R}^n$ can be gated by the positively homogeneous[2] function

$$\varphi_S(x) := \left(\prod_{i \in S} x_i \right)^{1/|S|}$$

This is quite useful for computing n-variable Boolean functions. In this case, $X := \mathbf{R}^n$, $Y = \mathbf{R}$, and the subset $\mathcal{K}$ of input-output patterns is $\{0,1\}^n \times \{0,1\}$ (and, in the fuzzy case, $[0,1]^n \times [0,1]$). We shall prove that the neural network

$$y = \sum_{S \in 2^N} w^S \left(\prod_{i \in S} x_i \right)^{1/|S|}$$

can compute any Boolean function b defined on $\{0,1\}^n$, in the sense that there exist weights w^S such that

$$\forall\, x \in \{0,1\}^n, \quad \sum_{S \in 2^N} w^S \left(\prod_{i \in S} x_i \right)^{1/|S|} = b(x)$$

This allows us to extrapolate them to *fuzzy statements* $x \in [0,1]^n$. Another (obvious) choice of preprocessing functions having the same property is given by the multilinear function φ_S defined by

$$\varphi_S(x) := \prod_{i \in S} x_i \prod_{j \notin S} (1 - x_j)$$

because we can always write

$$\forall\, x \in \{0,1\}^n, \quad b(x) = \sum_{y \in \{0,1\}^n} b(y) \prod_{\{i | y_i = 1\}} x_i \prod_{\{j | y_j = 0\}} (1 - x_j)$$

This is the analytical version of the standard result from Boolean algebra stating that any Boolean function may be expressed in the *disjunctive*

[2]The choice of positively homogeneous functions allows independence of the normalization rule attributing the value 1 to "true."

normal form. The drawback is that such functions φ_S involve the afferent signals from presynaptic neurons that do not belong to S, contrary to the first example. However, there is nothing to "compute," because we have an explicit formula $w^S = b(y_S)$, where y_S denotes the characteristic function of the subset S. □

3. Nonlinear Automata

The next class of rules of propagation is made of maps f of the form

$$f_j(x, W) := g_j\left(\sum_{S\in 2^N} w_j^S \varphi_S(x)\right)$$

where g_j are automata of various forms.

McCulloch-Pitts Neurons These are also called *threshold logic units*. They are associated with functions g_j that are built from the Heaviside function $\mathbf{1}$ defined by $\mathbf{1}(\lambda) = 0$ if $\lambda < 0$, and $\mathbf{1}(\lambda) = 1$ if $\lambda \geq 0$. The rules of propagation require also *thresholds* β_j. Therefore, such a neural network evolves according to the law

$$y_j = \begin{cases} 1 & \text{if } \sum_{S\in 2^N} w_j^S \varphi_S(x) \geq \beta_j \\ 0 & \text{if } \sum_{S\in 2^N} w_j^S \varphi_S(x) < \beta_j \end{cases}$$

Naturally, we can replace the Heaviside function $\mathbf{1}$ by functions g such that $g(x) := A$ when $x \geq 0$, and $g(x) = a$ when $x < 0$.

"Continuous" Automata Because the Heaviside function $\mathbf{1}$ and the forgoing two-valued functions g are not continuous, it may be useful to replace them by continuous and even differentiable approximations in some problems (Figure 1.5). This can be done by using the function $g_{k,\gamma}$:

$$g_{k,\gamma}(\lambda) := \frac{A\gamma e^{k\lambda} + a}{\gamma e^{k\lambda} + 1}$$

where $A, a > 0, \gamma$ and $k > 0$ are given parameters representing the automaton. We set $g_k := g_{k,1}$. [the constant $\beta := -\log\gamma/k$ represents a threshold because $g_{k,\gamma}(\lambda) = g_k(\lambda - \beta)$]. The function $g_{k,\gamma}$ maps $\mathbf{R}$ onto $[a, A]$ and has a "sigmoid" shape. The parameter $T := 1/k$ is called the "temperature," by analogy with spin glasses. Its derivative is equal to $k\gamma e^{k\lambda}(A - a)/(\gamma e^{k\lambda} + 1)^2$.

We observe that when k goes to ∞, the function g_k converges to the