LEARNING ALGORITHMS FOR NEURAL NETWORKS

Thesis by

Amir Atiya

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1991

ACKNOWLEDGEMENTS

I wish to express my gratitude to my advisor Professor Yaser Abu-Mostafa for all the guidance and advice he has accorded in the course of this work. I appreciate especially the encouragement and friendship he has extended to me. I would like to thank Dr. Pierre Baldi. The fourth chapter of my thesis is a result of a collaborative effort with Dr. Baldi. Working with Dr. Baldi has been immensely helpful and stimulating. I would like to extend my gratitude for the useful discussions and help to my fellow graduate students: Tzi-Dar Chiueh, Anamitra Makur, Chuanyi Ji, Yiu-Fai Wong, Dr. Robert Snapp, Alan Yamamura, Timothy Brown, Ruth Erlanson, Nabeel Riza, John Miller, and Mark Neifeld. I am thankful to the Institute for such a stimulating educational environment, and for the financial support. I gratefully acknowledge my parents for their constant encouragement.

ABSTRACT

This thesis deals mainly with the development of new learning algorithms and the study of the dynamics of neural networks. We develop a method for training feedback neural networks. Appropriate stability conditions are derived, and learning is performed by the gradient descent technique. We develop a new associative memory model using Hopfield's continuous feedback network. We demonstrate some of the storage limitations of the Hopfield network, and develop alternative architectures and an algorithm for designing the associative memory. We propose a new unsupervised learning method for neural networks. The method is based on applying repeatedly the gradient ascent technique on a defined criterion function. We study some of the dynamical aspects of Hopfield networks. New stability results are derived. Oscillations and synchronizations in several architectures are studied, and related to recent findings in biology. The problem of recording the outputs of real neural networks is considered. A new method for the detection and the recognition of the recorded neural signals is proposed.

Table of Contents

INTRODUCTION

Because of the clear superiority of the brain over conventional computers in many information processing tasks, attention has focused during the previous decade on developing computing structures, called neural networks, which emulate the structure and the functionality of the brain. The study of neural networks has the dual purpose of using our knowledge about the brain to develop efficient solutions to difficult real world computational problems, as well as shedding some light on the possible information processing mechanisms of the brain.

A neural network is a massively parallel network of interconnected analog processing elements (neurons) (see Fig. 0.1 for an example). The parallel nature of the neural network enables it to achieve potentially computation speeds not attainable by conventional sequential computers. The function of an individual processing element is very simple, and is often taken as the weighted sum of its inputs, passed through a non-linear function, usually taken as sigmoid-shaped. The weights of the neurons are the parameters, which define the functionality of the network. The motivation behind choosing the neuron functions as given was initially biological, but also turned out to be very practical for various applications and for VLSI implementations. The real neurons in the brain communicate by means of trains of impulses (or spikes). It is believed that the short-time average spike rate is one of the most relevant information processing parameters. Biological neurons are also densely interconnected, and the output of each neuron affects the output of each neuron connected to it with a particular "efficacy". The outputs of the neurons in the described "artificial" neural network represent the spike rates, whereas the weights represent the efficacies.

One of the powerful aspects of the neural network is that it can perform complex non-linear mappings. Some architectures exhibit rich dynamical properties,

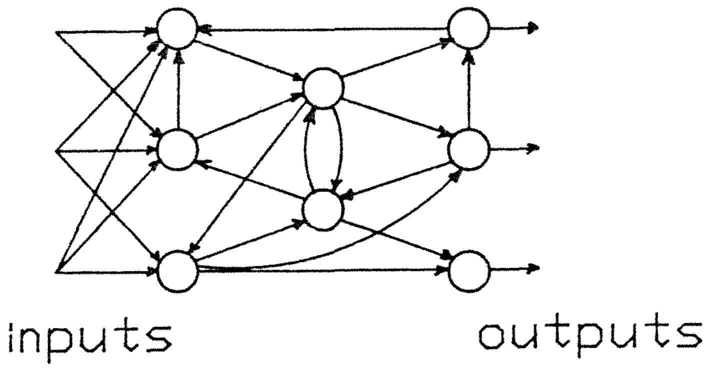

Fig. 0.1: An example of a neural network.

suggesting applications such as signal processing, robotics, and control. The ability to learn is one of the main advantages that make neural networks so attractive. Efficient learning algorithms have been devised to determine the weights of the network, according to the data of the computational task to be performed. Learning in neural networks has also been inspired by biological systems. The efficacies of real neurons vary slowly with time according to the given stimuli, thus accounting for the long term memory. The learning ability of the neural network makes it suitable for problems whose structure is relatively unknown, such as pattern recognition, medical diagnosis, time series prediction, control, and others. Such real world problems cannot be easily captured by a simple mathematical formula or algorithm. Therefore, the way to model a problem is to consider simply input/output examples of the process (called training patterns). A system which has the capability of learning by examples (for example, a neural network) would in general be suitable for such kinds of tasks.

In this thesis we will study the dynamics and capabilities of neural networks, develop learning algorithms for various architectures and purposes, and study some biological aspects and insights. In Chapter 1 we briefly review two main architectures of neural networks: feedforward and feedback networks, and we review the method for learning by examples for feedforward networks. The main purpose of this section is to develop a new method for learning by examples for feedback networks. In Chapter 2 we will study the storage capabilities and limitations of feedback associative memories. We propose a network architecture, and develop a learning algorithm for the storage of real-valued vectors. In Chapter 3 we develop a neural network method for unsupervised learning. The work in Chapter 4, done with the collaboration of Dr. Pierre Baldi of Jet Propulsion Laboratory, deals with the study of some of the dynamical aspects of feedback neural networks. Oscillations in neural networks are studied, and biological implications are proposed. In Chapter 5 we consider the problem of recording the outputs of real neurons. We develop new signal processing techniques for the accurate detection of the neural activity.

CHAPTER 1

LEARNING IN FEEDBACK NEURAL NETWORKS

1.1 FEEDFORWARD NETWORKS

There are mainly two categories of neural network architectures: feedforward and feedback networks. Feedforward (or multi-layer) networks have been extensively invesigated in the neural networks literature (see Lippmann's review on neural networks [1], and see Wasserman [2]). They consist of several layers connected as shown in Fig. 1.1. The last layer is called the *output* or *visible layer*, and the other layers are called *hidden layers*. In most applications one usually uses one or two hidden layers. The number of neurons per hidden layer depends on the problem considered. The usual way is to specify it by trial and error, and of course the more difficult the problem, the larger the required sizes of the hidden layers. Let $y_i^{(l)}$ denote the output of the i^{th} neuron of layer l ($y_i^{(0)}$ denotes the i^{th} input to the network). The function of the network is given by

$$y_i^{(l)} = f\big[\sum_{j=1}^{N_{l-1}} w_{ij}^{(l,l-1)} y_j^{(l-1)} + \theta_i^{(l)}\big], \qquad l = 1, ..., L, \quad i = 1, ..., N_l,$$

where $w_{ij}^{(l,l-1)}$ denotes the *weight* from neuron j of layer $l-1$ to neuron i of layer l, $\theta_i^{(l)}$ is the *threshold* of neuron i of layer l. The function f is taken as a unit step or a sign function (see Figs. 1.2 and 1.3), or as a sigmoid-shaped function, e.g.,

$$f(x) = \tanh(x), \tag{1.1}$$

(see Fig. 1.4), or

$$f(x) = \frac{1}{1+e^{-x}} \tag{1.2}$$

(see Fig. 1.5). Usually, one uses one of the previous two forms because of their smoothness.

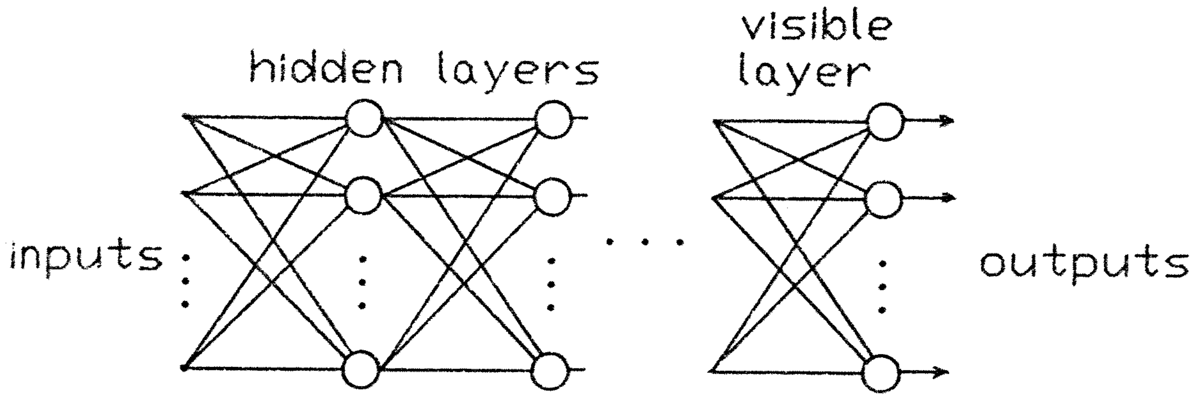

Fig. 1.1: A feedforward network.

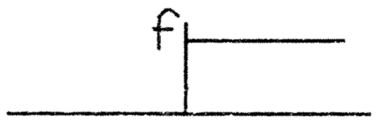

Fig. 1.2: A unit-step neuron function.

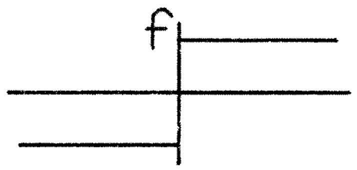

Fig. 1.3: A sign neuron function.

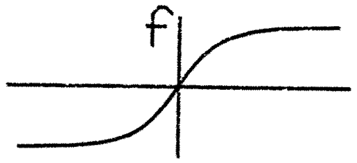

Fig. 1.4: A sigmoid neuron function from -1 to 1.

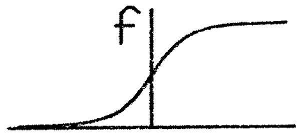

Fig. 1.5: A sigmoid neuron function from 0 to 1.

The breakthrough for multi-layer networks was the development of a method for learning by examples, the backpropagation learning algorithm, which was developed by Werbos [3] and Rumelhart *et al.* [4]. The backpropagation algorithm has made it possible to design multi-layer networks for numerous applications, such as adaptive control [5], sonar [6], stock market prediction [7], speech recognition [8]. The method is described as follows.

Let $\mathbf{x}(1), ..., \mathbf{x}(M)$ be the given input vectors, and $\mathbf{y}(1), ..., \mathbf{y}(M)$ be the corresponding desired output vectors. It is required to adjust the weights and thresholds of the network, so that the network ultimately maps each $\mathbf{x}(m)$ to $\mathbf{y}(m)$ as closely as possible. We define the sum of square error:

$$E = \sum_{m=1}^{M} E(m),$$

$$E(m) = ||\mathbf{y}(m) - \mathbf{y}^{(L)}||^2,$$

where $\mathbf{y}^{(L)}$ represents the vector of outputs of the network, when the input is $\mathbf{x}(m)$. In the backpropagation algorithm we initialize all the weights and thresholds to small, random values. Then, we iteratively adjust the weights and the thresholds in a steepest descent manner; i.e.,

$$w_{ij}^{(l,l-1)}(t+1) = w_{ij}^{(l,l-1)}(t) - \rho \frac{\partial E}{\partial w_{ij}^{(l,l-1)}},$$

$$\theta_i^{(l)}(t+1) = \theta_i^{(l)}(t) - \rho \frac{\partial E}{\partial \theta_i^{(l)}},$$

where ρ is the step size. Another variation, which is in some ways faster and therefore more often used is to adjust the weights after presenting each example, instead of after cycling through the whole set of examples. Therefore, in this variation we present the input examples cyclically, and each time we adjust the weights in a steepest descent on $E(m)$, as follows,

$$w_{ij}^{(l,l-1)}(t+1) = w_{ij}^{(l,l-1)}(t) - \rho \frac{\partial E(m)}{\partial w_{ij}^{(l,l-1)}},$$

$$\theta_i^{(l)}(t+1) = \theta_i^{(l)}(t) - \rho \frac{\partial E(m)}{\partial \theta_i^{(l)}}.$$

The procedure in detail, which is called the delta update rule [4], is as follows. We start by adjusting the weights of the last layer and work backwards until the first hidden layer. We adjust the weights according to

$$w_{ij}^{(l,l-1)}(t+1) = w_{ij}^{(l,l-1)}(t) + \rho \delta_i^{(l)} y_j^{(l-1)},$$

$$\theta_i^{(l)}(t+1) = \theta_i^{(l)}(t) + \rho \delta_i^{(l)},$$

where $\delta_i^{(l)}$ is an error term for neuron i of layer l, calculated as follows: For the last layer,

$$\delta_j^{(L)} = f'\Big[\sum_{k=1}^{N_{L-1}} w_{jk}^{(L,L-1)} y_k^{(L-1)} + \theta_j^{(L)}\Big](y_j(m) - y_j^{(L)}).$$

We propagate backwards, to calculate the $\delta_j^{(l)}$'s for the remaining layers:

$$\delta_j^{(l-1)} = f'\Big[\sum_{k=1}^{N_{l-2}} w_{jk}^{(l-1,l-2)} y_k^{(l-2)} + \theta_j^{(l-1)}\Big]\Big(\sum_{k=1}^{N_l} \delta_k^{(l)} w_{kj}^{(l,l-1)}\Big).$$

Although the backpropagation algorithm is guaranteed to converge to only a local rather than a global minimum of the error function, experience has shown that in many problems it, in fact, converges to the global minimum. The algorithm has proven its effectiveness and usefulness in many applications.

1.2 FEEDBACK NETWORKS

Feedback networks have received a lot of attention in the neural networks literature. The feedback network consists of N neurons; the output of each is fed back to all other neurons via weights w_{ij}. The operation of the network is given in terms of defined dynamics, which describes the time evolution of the neuron outputs. The system's evolution can be expressed in a discrete-time or continuous-time formulation. In the discrete-time formulation (introduced by Hopfield [9]), the evolution equation is given by

$$y_i' = f(\sum_{i=1}^{N} w_{ij} y_j + \theta_i),$$

where w_{ij} is the weight from neuron i to neuron j, θ_i is the threshold for neuron i, y_i is the state for neuron i, y_i' is the new state for neuron i, and f is either a sign function or a sigmoid function (most researchers consider f as a sign function).

The continuous-time formulation was developed by Amari [10], Grossberg and Cohen [11], and Hopfield [12]. It is given by the set of differential equations (called the Hopfield continuous model):

$$\tau_i \frac{du_i}{dt} = -u_i + \sum_{i=1}^{N} w_{ij} f(u_j) + \theta_i \equiv g_i(u_1, ..., u_N), \qquad (1.3)$$

$$y_i = f(u_i),$$

where, again, w_{ij} is the weight from neuron i to neuron j, θ_i is the threshold for neuron i, τ_i is the time constant for neuron i, and f is a sigmoid-shaped function, e.g., (1.1) or (1.2).

1.3 DYNAMICS AND STABILITY

Unlike feedforward networks, feedback networks can produce time-varying outputs. They could produce periodic, or chaotic motion, or possibly, eventually constant outputs. They possess, therefore, richer types of behavior, and hence richer information processing capabilities.

In this thesis we will concentrate mainly on the Hopfield continuous model. Before considering some of its dynamical aspects, the following relevant stability concepts are reviewed. We will provide only heuristic explanations of some of the concepts. The rigorous definitions can be found in Vidyasagar [13].

An *equilibrium* of a system (consider the system (1.3)) is a point $\mathbf{u}^* = (u_1^*, ..., u_N^*)^T$ satisfying $g_i(u_1^*, ..., u_N^*) = 0, \ i = 1, ..., N$.

An equilibrium $\mathbf{u}^*$ is said to be *asymptotically stable* if:

1) It is possible to force any trajectory starting from some open neighborhood U around the equilibrium to remain as close as desired to the equilibrium $\mathbf{u}^*$ for all $t > T$ by choosing U sufficiently small.

2) The trajectory goes to the equilibrium $\mathbf{u}^*$ as t goes to infinity.

An equilibrium that does not satisfy 1) is said to be an *unstable equilibrium.*

A system is said to be *globally stable* (or simply *stable*) if every trajectory converges to an equilibrium.

A system is *globally asymptotically stable* if every trajectory converges to a unique equilibrium.

There has been considerable interest in studying stability properties for continuous neural networks. For instance, Hopfield [12] and Cohen and Grossberg [11] showed that a network with symmetric weight matrix and monotonically increasing functions f is stable. The proof is by constructing an *energy* or a *Lyapunov function*, that is, a function which is non-increasing along the trajectories (see [13]). The local minima of such a function correspond to the equilibria of the network. Further results and discussions on stability issues will be presented in the next section, and in Chapter 4.

1.4 A LEARNING ALGORITHM FOR FEEDBACK NETWORKS

Feedback neural networks have been shown to be quite successful in performing a number of computational tasks. Examples of applications are the traveling salesman problem (Hopfield and Tank [14]), A/D conversion and linear programming (Tank and Hopfield [15]), image restoration (Zhou *et al.* [16]), associative memories (Hopfield [9]) (associative memories are an important class of applications, to be discussed in detail in the next chapter). The network for solving such problems is usually specified by constructing an energy function whose minimum corresponds to the solution of the problem. This energy function specifies the weight matrix of the network. Iterating this network leads to the convergence to the (global, it is hoped) minimum of the energy function and hence to the solution of the problem. This design technique, though elegant, is not a sufficiently systematic design method, and one is not guaranteed to find the required energy function for any given problem. We will propose here a method for learning by examples on general networks containing feedback as well as feedforward connections. In other words we will extend the backpropagation training algorithm to general networks containing feedback connections (also see Atiya [17]).

Consider a group of N neurons that could be fully interconnected (see Fig. 1.6 for an example). The weight matrix $\mathbf{W}$ can be asymmetric. The inputs are also weighted before entering into the network (let $\mathbf{V}$ be the weight matrix). Let $\mathbf{x}$ and $\mathbf{y}$ be the input and output vectors, respectively. Our model is governed by the following set of differential equations (Hopfield [12]):

$$\tau \frac{d\mathbf{u}}{dt} = -\mathbf{u} + \mathbf{W}\mathbf{f}(\mathbf{u}) + \mathbf{V}\mathbf{x} + \theta, \qquad \mathbf{y} = \mathbf{f}(\mathbf{u}), \tag{1.4}$$

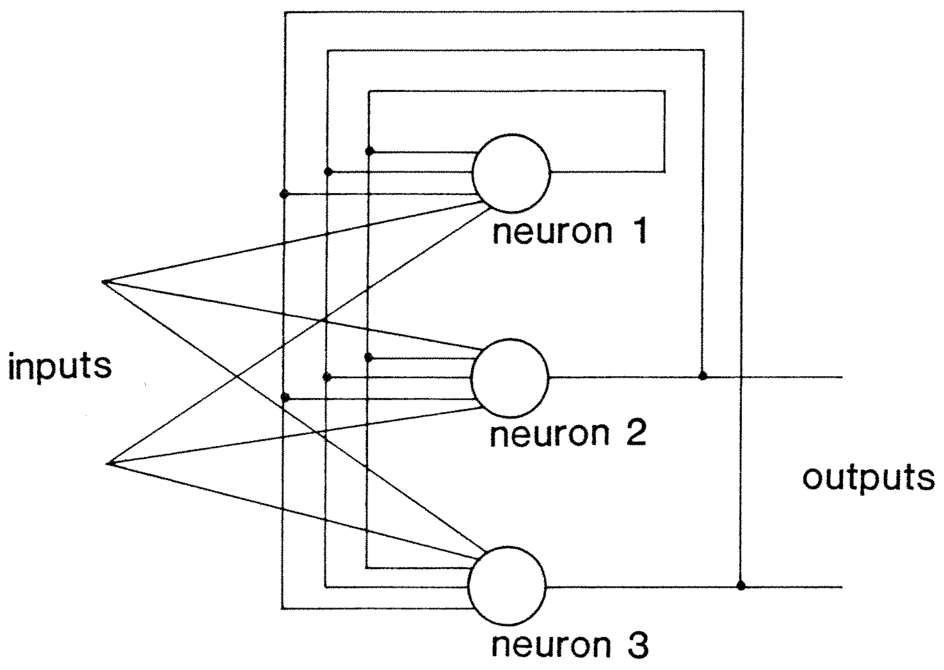

Fig. 1.6: A feedback network.

where $\mathbf{f}(\mathbf{u}) = (f(u_1), ..., f(u_n))^T$, ^{T} denotes the transpose operator, f is a sigmoid function or any other bounded and differentiable function, τ is the time constant, and $\theta = (\theta_1, ..., \theta_N)^T$ is the threshold vector.

In developing a learning algorithm for feedback networks, one has to pay attention to the following. The state of the network evolves in time until it goes to an equilibrium, or possibly other types of behavior such as periodic or chaotic motion could occur. However, we are interested in having a steady and fixed output for every input applied to the network. Therefore, we have the following two important requirements for the network. Starting any initial condition, the state should ultimately go to an equilibrium. The other requirement is that we have to have a unique equilibrium state for a fixed input. It is, in fact, that equilibrium state that determines the final output. The objective of the learning algorithm is to adjust the parameters (weights) of the network in small steps, so as to move the unique equilibrium state in a way that will result finally in an output as close as possible to the required one (for each given input). The existence of more than one equilibrium state for a given input causes the following problems. In some iterations one might be updating the weights so as to move one of the equilibrium states in a sought direction, while in other iterations a different equilibrium state is moved. Another important point is that when implementing the network (after the completion of learning), for a fixed input there can be more than one possible output, depending on what the initial condition is. This leads us to look for a condition on the matrix $\mathbf{W}$ to guarantee the global asymptotic stability of the system.

Theorem: A network (not necessarily symmetric) satisfying either

a) $\sum_i \sum_j w_{ij}^2 < 1/\max(f')^2$,

or

b) $\sum_j |w_{ij}| < 1/\max(f')$, all i,

exhibits no other behavior except going to a unique equilibrium for a given input.

Proof: Consider Case a) first. Let $\mathbf{u}_1(t)$ and $\mathbf{u}_2(t)$ be two trajectories of (1.4) starting from two different initial states. Let

$$J(t) = \|\mathbf{u}_1(t) - \mathbf{u}_2(t)\|^2,$$

where $\| \ \|$ is the two-norm. Differentiating J with respect to time, we obtain

$$\frac{dJ(t)}{dt} = 2(\mathbf{u}_1(t) - \mathbf{u}_2(t))^T \Big(\frac{d\mathbf{u}_1(t)}{dt} - \frac{d\mathbf{u}_2(t)}{dt}\Big).$$

Using (1.4) , the expression becomes

$$\frac{dJ(t)}{dt} = -\frac{2}{\tau}\|\mathbf{u}_1(t) - \mathbf{u}_2(t))\|^2 + \frac{2}{\tau}(\mathbf{u}_1(t) - \mathbf{u}_2(t))^T\mathbf{W}\big[\mathbf{f}(\mathbf{u}_1(t)) - \mathbf{f}(\mathbf{u}_2(t))\big].$$

Using Schwarz's Inequality, we obtain

$$\frac{dJ(t)}{dt} \leq -\frac{2}{\tau}\|\mathbf{u}_1(t) - \mathbf{u}_2(t)\|^2 + \frac{2}{\tau}\|\mathbf{u}_1(t) - \mathbf{u}_2(t)\| \cdot \|\mathbf{W}\left[\mathbf{f}(\mathbf{u}_1(t)) - \mathbf{f}(\mathbf{u}_2(t))\right]\|.$$

Again, by Schwarz's Inequality,

$$\mathbf{w}_i\left[\mathbf{f}(\mathbf{u}_1(t)) - \mathbf{f}(\mathbf{u}_2(t))\right] \leq \|\mathbf{w}_i\| \cdot \|\mathbf{f}(\mathbf{u}_1(t)) - \mathbf{f}(\mathbf{u}_2(t))\|, \qquad i = 1, ..., N \tag{1.5}$$

where $\mathbf{w}_i$ denotes the i^{th} row of $\mathbf{W}$. Using the mean-value theorem, we get

$$\|\mathbf{f}(\mathbf{u}_1(t)) - \mathbf{f}(\mathbf{u}_2(t))\| \leq (\max|f'|)\|\mathbf{u}_1(t) - \mathbf{u}_2(t)\|. \tag{1.6}$$

Using (1.5), (1.6), and the expression for $J(t)$, we get

$$\frac{dJ(t)}{dt} \leq -\alpha J(t), \tag{1.7}$$

where

$$\alpha = \frac{2}{\tau} - \frac{2}{\tau}(\max|f'|)\sqrt{\sum_i \sum_j w_{ij}^2}.$$

By Condition a) in the hypothesis of the Theorem, α is strictly positive. Multiplying both sides of (1.7) by $\exp(\alpha t)$, the inequality

$$\frac{d}{dt}(J(t)e^{\alpha t}) \leq 0$$

results, from which we obtain

$$J(t) \leq J(0)e^{-\alpha t}.$$

From that and from the fact that J is non-negative, it follows that $J(t)$ goes to zero as $t \rightarrow \infty$. Therefore, any two trajectories corresponding to any two initial conditions ultimately approach each other. To show that this asymptotic solution is in fact an equilibrium, we simply take $\mathbf{u}_2(t) = \mathbf{u}_1(t+T)$, where T is a constant, and applies the above argument (that $J(t) \rightarrow 0$ as $t \rightarrow \infty$), and hence $\mathbf{u}_1(t+T) \rightarrow \mathbf{u}_1(t)$ as $t \rightarrow \infty$ for any T, and this completes the first part of the proof.

Now, consider Case b). Again, consider two trajectories $\mathbf{u}_1(t) = (u_{11}(t), ..., u_{1N}(t))^T$, and $\mathbf{u}_2(t) = (u_{21}(t), ..., u_{2N}(t))^T$. Define

$$J_i(t) = \big(u_{1i}(t) - u_{2i}(t)\big)^2,$$

and

$$J(t) = \max_i J_i(t). \tag{1.8}$$

Let $I(t)$ be the set of indices at which the maximum in (1.8) is attained. Differentiating J, one obtains

$$\begin{aligned}
\frac{dJ(t)}{dt} &= 2(u_{1i}(t) - u_{2i}(t))\Big(\frac{du_{1i}(t)}{dt} - \frac{du_{2i}(t)}{dt}\Big),\\
&= \frac{2}{\tau}(u_{1i}(t) - u_{2i}(t))\Big[-(u_{1i}(t) - u_{2i}(t)) + \sum_j w_{ij}(f(u_{1j}(t)) - f(u_{2j}(t)))\Big],\\
&\leq -\frac{2}{\tau}(u_{1i}(t) - u_{2i}(t))^2 + \frac{2}{\tau}|u_{1i}(t) - u_{2i}(t)|\sum_j |w_{ij}||f(u_{1j}(t)) - f(u_{2j}(t))|,
\end{aligned}$$

where i denotes the element of $I(t)$, which has the largest $\frac{dJ_i(t)}{dt}$. Using the mean-value theorem, and using the fact that $|u_{1j}(t) - u_{2j}(t)| \leq |u_{1i}(t) - u_{2i}(t)|$, we get

$$\frac{dJ(t)}{dt} \leq -\alpha_i J(t),$$

where $\alpha_i = \frac{2}{\tau}\big[1 - \max|f'|\sum_j |w_{ij}|\big]$. Using an argument similar to that of the first part of the proof, we can conclude that under Condition b), the system converges to a unique equilibrium. ∎

For example, if the function f is of the widely used form, $f(u) = 1/(1+e^{-u})$, then Condition a) requires that the sum of the square of the weights be less than 16. Condition b) requires that the sum of the absolute values of the weights converging onto each neuron be less than 4. Note that for any function f, scaling does not have an effect on the overall results. We have to work in our updating scheme subject to one of the constraints given in the Theorem. We note that a condition for global asymptotic stability very similar to Condition b) was also independently derived by Hirsch [18].

1.5 LEARNING

Consider a fully connected network, as in Fig. 1.6. The neurons from 1 to K are output neurons, and the neurons from $K+1$ to N are hidden neurons. Let $(\mathbf{x}(m), \mathbf{y}(m)),\ m = 1, ..., M$ be the input/output vector pairs of the function to be implemented. We would like to minimize the sum of the square error, given by

$$E = \sum_{m=1}^{M} E(m),$$

where

$$E(m) = \sum_{i=1}^{K} (y_i - y_i(m))^2, \tag{1.9}$$

and $\mathbf{y} = (y_1, ..., y_N)^T$ is the network (equilibrium) output vector when the input is $\mathbf{x}(m)$. The learning process is performed by feeding the input examples $\mathbf{x}(m)$ sequentially to the network, each time updating the weights in an attempt to minimize the error.

We would like each iteration to update the weight matrices $\mathbf{W}$ and $\mathbf{V}$, and the threshold vector θ, so as to move the equilibrium in a direction to decrease the error. We need therefore to know the change in the error produced by a small change in the weight matrices. Let $\frac{\partial E(m)}{\partial \mathbf{W}}$ and $\frac{\partial E(m)}{\partial \mathbf{V}}$ denote the matrices whose $(i,j)^{th}$ element are $\frac{\partial E(m)}{\partial w_{ij}}$ and $\frac{\partial E(m)}{\partial v_{ij}}$, respectively, and let $\frac{\partial E(m)}{\partial \theta}$ be the column vector whose i^{th} element is $\frac{\partial E(m)}{\partial \theta_i}$.

Assume that the time constant τ is sufficiently small so as to allow the network to settle quickly to the equilibrium state, which is given by the solution of the equation:

$$\mathbf{y} = \mathbf{f}(\mathbf{W}\mathbf{y} + \mathbf{V}\mathbf{x} + \theta). \tag{1.10}$$

Differentiating (1.10), we obtain

$$\frac{\partial y_j}{\partial w_{kp}} = f'(z_j)(\sum_n w_{jn}\frac{\partial y_n}{\partial w_{kp}} + y_p\delta_{jk}), \qquad k,p = 1,...,N,$$

where

$$\delta_{jk} = \begin{cases} 1 & \text{if } j = k, \\ 0 & \text{otherwise,} \end{cases}$$

and

$$z_j = \sum_n w_{jn}y_n + \sum_n v_{jn}x_n(m) + \theta_j.$$

We can write in vector notation as

$$\frac{\partial \mathbf{y}}{\partial w_{kp}} = (\mathbf{\Lambda} - \mathbf{W})^{-1}\mathbf{b}^{kp}, \tag{1.11}$$

where $\mathbf{\Lambda} = \text{diag}(1/f'(z_i))$ and $\mathbf{b}^{kp}$ is the N-dimensional vector whose i^{th} component is given by

$$b_i^{kp} = \begin{cases} y_p & \text{if } i = k, \\ 0 & \text{otherwise.} \end{cases}$$

By the chain rule,

$$\frac{\partial E(m)}{\partial w_{kp}} = \sum_j \frac{\partial E(m)}{\partial y_j}\frac{\partial y_j}{\partial w_{kp}},$$

which, upon substituting from (1.11), can be put in the form $y_p\mathbf{g}_k^T\frac{\partial E(m)}{\partial \mathbf{y}}$, where $\mathbf{g}_k$ is the k^{th} column of $(\mathbf{\Lambda} - \mathbf{W})^{-1}$. Finally, we obtain the required expression, which is

$$\frac{\partial E(m)}{\partial \mathbf{W}} = \left[\mathbf{\Lambda} - \mathbf{W}^T\right]^{-1}\frac{\partial E(m)}{\partial \mathbf{y}}\mathbf{y}^T,$$

where $\frac{\partial E(m)}{\partial \mathbf{y}}$ is obtained by differentiating (1.9). We get

$$\frac{\partial E(m)}{\partial \mathbf{y}} = 2(y_1 - y_1(m), ..., y_K - y_K(m), 0, ..., 0)^T. \tag{1.12}$$

Regarding $\frac{\partial E(m)}{\partial \mathbf{V}}$, it is obtained by differentiating (1.10) with respect to v_{kp}. We get similarly

$$\frac{\partial \mathbf{y}}{\partial v_{kp}} = (\mathbf{\Lambda} - \mathbf{W})^{-1}\mathbf{c}^{kp}$$

where $\mathbf{c}^{kp}$ is the N-dimensional vector whose i^{th} component is given by

$$c_i^{kp} = \begin{cases} x_p(m) & \text{if } i = k \\ 0 & \text{otherwise.} \end{cases}$$

A derivation very similar to the case of $\frac{\partial E(m)}{\partial \mathbf{W}}$ results in the following required expression:

$$\frac{\partial E(m)}{\partial \mathbf{V}} = \left[\mathbf{\Lambda} - \mathbf{W}^T\right]^{-1} \frac{\partial E(m)}{\partial \mathbf{y}} \mathbf{x}(m)^T.$$

Similarly, we get also

$$\frac{\partial E(m)}{\partial \theta} = \left[\mathbf{\Lambda} - \mathbf{W}^T\right]^{-1} \frac{\partial E(m)}{\partial \mathbf{y}}.$$

As in the backpropagation method, we update the weights here by the steepest descent method. We use the variant discussed in Section 1.1, where we present the examples in a cyclical manner, and updates the weights after each presentation according to the gradient of $E(m)$. This becomes in our case

$$\mathbf{W}(t+1) = \mathbf{W}(t) - \rho\left[\mathbf{\Lambda} - \mathbf{W}^T\right]^{-1} \frac{\partial E(m)}{\partial \mathbf{y}} \mathbf{y}^T,$$

$$\mathbf{V}(t+1) = \mathbf{V}(t) - \rho\left[\mathbf{\Lambda} - \mathbf{W}^T\right]^{-1} \frac{\partial E(m)}{\partial \mathbf{y}} \mathbf{x}(m)^T,$$

$$\theta(t+1) = \theta(t) - \rho\left[\mathbf{\Lambda} - \mathbf{W}^T\right]^{-1} \frac{\partial E(m)}{\partial \mathbf{y}},$$

where $\frac{\partial E(m)}{\partial \mathbf{y}}$ is given by (1.12), and ρ is the step size. Updating the weights is subject to any of the constraints given in the Theorem. If at some iteration the weights violate one of the constraints, then they are scaled so as to project them back onto the surface given by the constraint (simulations have shown that the constraint a) in general gives better results).

We assert that the matrix $\mathbf{\Lambda} - \mathbf{W}^T$, whose inverse appears in the update equations, can never be singular. We prove that as follows. By Gershgorin's Theorem

[19], the eigenvalues of $\mathbf{\Lambda} - \mathbf{W}^T$ lie in the union of the circles (in the complex space) defined by

$$|\lambda - \frac{1}{f'(z_i)} + w_{ii}| \leq \sum_{j \neq i} |w_{ij}|.$$

Hence, if $\max(1/f') > \sum_{j=1}^{N} |w_{ij}|$, all i, then all eigenvalues of $\mathbf{\Lambda} - \mathbf{W}^T$ will have positive real parts. This previous condition is equivalent to condition b) given in the Theorem. Hence the matrix $\mathbf{\Lambda} - \mathbf{W}^T$ is guaranteed to be non-singular when enforcing condition b) during the update iterations. It can be also proven that if condition a) of the Theorem is enforced, then the matrix $\mathbf{\Lambda} - \mathbf{W}^T$ is non-singular.

One of the drawbacks of the steepest descent minimization method is its susceptibility to local minima. A way to reduce this is to add noise in the update equations to allow it to escape local minima. We have implemented a stochastic descent method, described as

$$\mathbf{W}(t+1) = \mathbf{W}(t) - \rho \frac{\partial E(m)}{\partial \mathbf{W}} + \alpha E(m)\mathbf{R},$$

$$\mathbf{V}(t+1) = \mathbf{V}(t) - \rho \frac{\partial E(m)}{\partial \mathbf{V}} + \alpha E(m)\mathbf{R},$$

$$\theta(t+1) = \theta(t) - \rho \frac{\partial E(m)}{\partial \theta} + \alpha E(m)\mathbf{R},$$

where $\mathbf{R}$ is a noise matrix whose elements are characterized by independent zero-mean, unity-variance Gaussian densities, and α is a parameter. Note that the control parameter is taken to be $E(m)$. Hence, the variance of the added noise tends to decrease the more we approach the ideal zero-error solution. This makes sense because for a large error, i.e., for an unsatisfactory solution, it pays more to add noise to the weight matrices in order to escape local minima. On the other hand, if the error is small, then we are possibly near the global minimum or near an acceptable solution, and hence we do not want too much noise in order not to be thrown out of that basin. Note that once we reach the ideal zero-error solution, the added noise as well as the gradient of $E(m)$ becomes zero for all m, and hence the increments of the weight matrices become zero, and the algorithm converges. This method was implemented on several examples. We have observed that it frequently converged to a zero-error solution when the original steepest descent method did not. Also, in general it was faster than the steepest descent method (in some examples more than twice as fast).

We wish to note that Almeida [20], and Pineda [21] have also independently developed a learning algorithm for feedback networks. In particular, Pineda's algorithm has some similarities to our method.

1.6 GENERALIZATION

In many cases where a large network is necessary, the constraints given by the Theorem might be too restrictive. Therefore we propose a general network, explained as follows. The neurons are partitioned into several groups (see Fig. 1.7 for an example). Within each group there are no restrictions on the connections and therefore the group could be fully interconnected (i.e., it could have feedback connections). The groups are connected to each other, but in such a way that there are no loops. The inputs to the whole network can be connected to the inputs of any of the groups (each input can have several connections to several groups). The outputs of the whole network are taken to be the outputs (or part of the outputs) of a certain group, say group f. The constraint given in the Theorem is applied on each intra-group weight matrix separately. The error function to be minimized is

$$E = \sum_{m=1}^{M} E(m),$$

where

$$E(m) = \sum_{i=1}^{K} (y_i^f - y_i(m))^2,$$

and $\mathbf{y}^f$ is the output vector of group f upon giving input $\mathbf{x}(m)$.

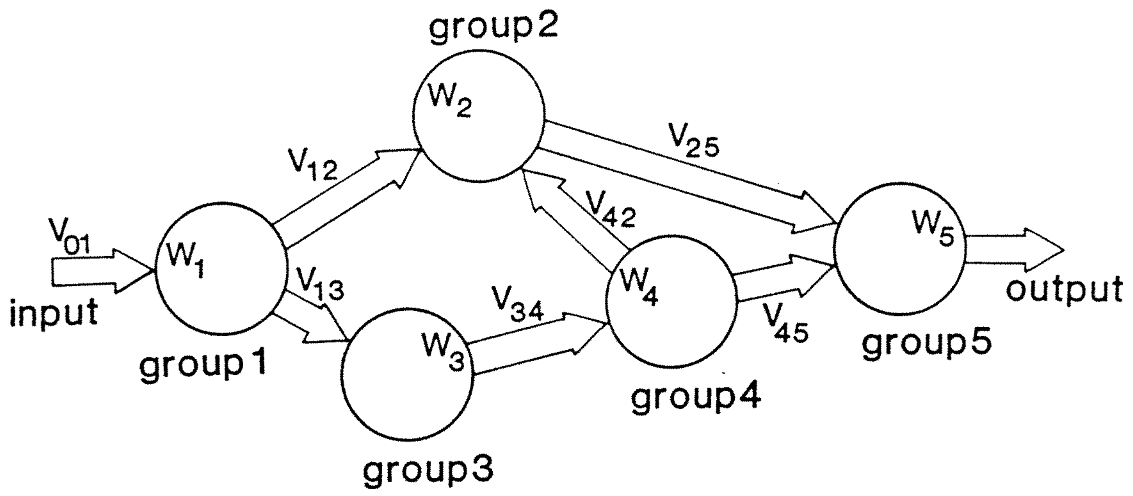

Fig. 1.7: An example of a general network (each group represents a feedback network).

We will also apply the steepest descent method. The gradients are evaluated as follows. Consider a single group l. Let $\mathbf{W}^l$ be the intra-group weight matrix of group l, $\mathbf{V}^{rl}$ be the matrix of weights between the outputs of group r and the inputs of group l, θ^l is the threshold vector of group l, and $\mathbf{y}^l$ be the output vector of group l. Let the respective elements be w_{ij}^l, v_{ij}^{rl}, θ_i^l, and y_i^l. Furthermore, let N_l be the number of neurons of group l. The equilibrium is given by

$$\mathbf{y}^l = \mathbf{f}(\mathbf{W}^l\mathbf{y}^l + \sum_{r\epsilon A_l} \mathbf{V}^{rl}\mathbf{y}^r + \theta^l),$$

where A_l is the set of the indices of the groups connected to the inputs of group l. A derivation similar to the one in the previous section gives

$$\frac{\partial E(m)}{\partial \mathbf{W}^l} = \left[\mathbf{\Lambda}^l - (\mathbf{W}^l)^T\right]^{-1}\frac{\partial E(m)}{\partial \mathbf{y}^l}(\mathbf{y}^l)^T,$$

$$\frac{\partial E(m)}{\partial \mathbf{V}^{rl}} = \left[\mathbf{\Lambda}^l - (\mathbf{W}^l)^T\right]^{-1}\frac{\partial E(m)}{\partial \mathbf{y}^l}(\mathbf{y}^r)^T,$$

$$\frac{\partial E(m)}{\partial \theta^l} = \left[\mathbf{\Lambda}^l - (\mathbf{W}^l)^T\right]^{-1} \frac{\partial E(m)}{\partial \mathbf{y}^l},$$

where $\mathbf{\Lambda}^l$ is the diagonal matrix whose i^{th} diagonal element is $1/f'(z_i^l)$, $z_i^l = \sum_k w_{ik}^l y_k^l + \sum_r \sum_k v_{ik}^{rl} y_k^r + \theta_i^l$. The vector $\frac{\partial E(m)}{\partial \mathbf{y}^l}$ associated with group l can be obtained in terms of the vectors $\frac{\partial E(m)}{\partial \mathbf{y}^j}$, $j \epsilon B_l$, where B_l is the set of the indices of the groups connected to the outputs of group l, as follows. Let $\frac{\partial \mathbf{y}^j}{\partial \mathbf{y}^l}$, $j \epsilon B_l$ be the matrix whose $(k,p)^{th}$ element is $\frac{\partial y_k^j}{\partial y_p^l}$. The elements of $\frac{\partial \mathbf{y}^j}{\partial \mathbf{y}^l}$ can be obtained by differentiating the equation for the equilibrium for group j, as follows,

$$\frac{\partial y_k^j}{\partial y_p^l} = f'(z_k^j)(v_{kp}^{lj} + \sum_n w_{kn}^j \frac{\partial y_n^j}{\partial y_p^l}).$$

Hence,

$$\frac{\partial \mathbf{y}^j}{\partial \mathbf{y}^l} = (\mathbf{\Lambda}^j - \mathbf{W}^j)^{-1} \mathbf{V}^{lj}. \qquad (1.13)$$

Using the chain rule, we can write

$$\frac{\partial E(m)}{\partial \mathbf{y}^l} = \sum_{j \epsilon B_l} \left(\frac{\partial \mathbf{y}^j}{\partial \mathbf{y}^l}\right)^T \frac{\partial E(m)}{\partial \mathbf{y}^j}.$$

We substitute from (1.13) into the previous equation to complete the derivation by obtaining

$$\frac{\partial E(m)}{\partial \mathbf{y}^l} = \sum_{j \epsilon B_l} (\mathbf{V}^{lj})^T \left[(\mathbf{\Lambda}^j - (\mathbf{W}^j)^T\right]^{-1} \frac{\partial E(m)}{\partial \mathbf{y}^j}. \qquad (1.14)$$

For each iteration we begin by updating the weights of group f (the group containing the final outputs). For that group $\frac{\partial E(m)}{\partial \mathbf{y}}$ equals simply $2(y_1^f - y_1(m), ..., y_K^f - y_K(m), 0, ..., 0)^T)$. Then we move backwards to the groups connected to that group and obtain their corresponding $\frac{\partial E(m)}{\partial \mathbf{y}}$ vectors using (1.14), update the weights, and proceed in the same manner until we complete updating all the groups. Updating the weights is also performed using the steepest descent rule.

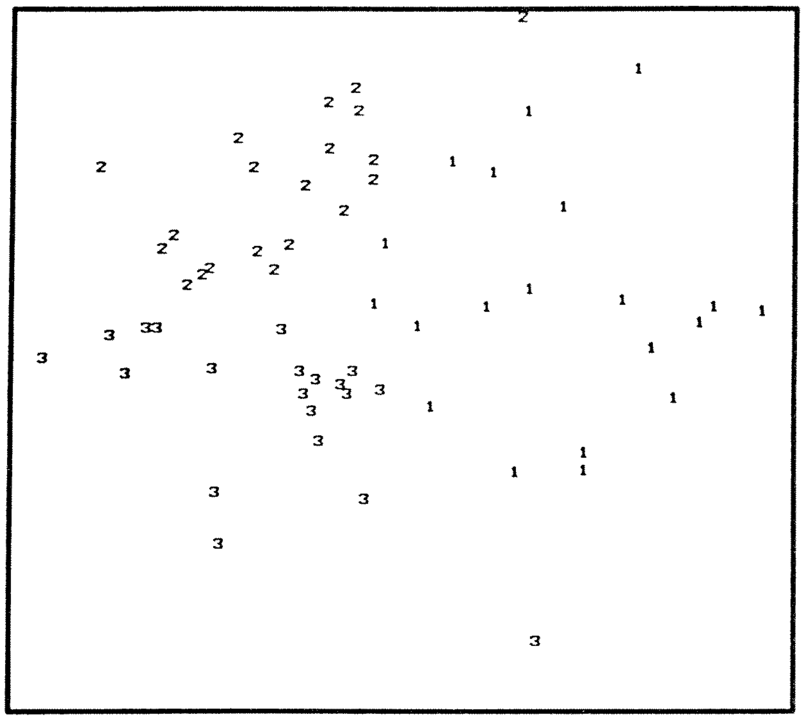

Fig. 1.8: A pattern recognition example.

1.7 AN IMPLEMENTATION EXAMPLE

A pattern recognition example is considered. Fig. 1.8 shows a set of two-dimensional training patterns from three classes. It is required to design a neural network recognizer with three output neurons. Each of the neurons should be on if a sample of the corresponding class is presented, and off otherwise; i.e., we would like to design a "winner-take-all" network. A single-layer three-neuron feedback network (fully connected except for self-connections w_{ii}) is implemented. We obtained 3.3% error. Performing the same experiment on a feedforward single-layer network with three neurons, we obtained a 20% error. For satisfactory results, a feedforward network should be two-layer. With one neuron in the first layer and three in the second layer, we got a 36.7% error. Finally, with two neurons in the first layer and three in the second layer, we got a match with the feedback case, with a 3.3% error.

1.8 CONCLUSION

A way to extend the backpropagation method to feedback networks has been proposed. Conditions on the weight matrix are obtained, to insure having a unique equilibrium, so as to prevent having more than one possible output for a fixed input. An algorithm for training the feedback network is developed. A general network structure for networks is also presented, in which the network consists of a number of feedback groups connected to each other in a feedforward manner. The method is applied to a pattern recognition example. With a single-layer feedback network it obtained good results. In general, we have observed that training feedback networks is somewhat slower and more susceptible to local minima than training feedforward networks. Since feedback networks are more general and hence more capable than feedforward networks, we propose the following strategy. We start by training a feedforward network until it converges, and then allow feedback connections within the layers and use the algorithm we have developed, to obtain a smaller error.

REFERENCES

[1] R. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, vol. 4, no. 2, pp. 4-22, 1987.

[2] Wasserman, *Neural Computing*, Van Nostrand Reinhold, New York, NY, 1989.

[3] P. Werbos, "Beyond regression: New tools for prediction and analysis in behavioral sciences," Harvard University dissertation, 1974.

[4] D. Rumelhart, G.Hinton, and R. Williams, "Learning internal representations by error propagation," in D. Rumelhart, J. McLelland and the PDP research

group (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition*, Vol. 1, MIT Press, Cambridge, MA, 1986.

[5] K. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, Vol. 1, No. 1, pp. 4-27, 1990.

[6] R. Gorman and T. Sejnowski, "Learned classification of sonar targets using a massively parallel network," *IEEE Trans. Acoust., Speech and Signal Processing*, Vol. 36, No. 7, pp. 1135-1140, 1988.

[7] T. Kimoto, K. Asakawa, M. Yoda, and M. Takeoka, "Stock market prediction system with modular neural networks," *Proceedings IJCNN Conference*, pp. I-1-I-6, San Diego, CA, June 1990.

[8] K. Lang, A. Waibel, and G. Hinton, "A time-delay neural network architecture for isolated word recognition," *Neural Networks*, Vol. 3, No. 1, pp. 23-43, 1990.

[9] J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. USA*, vol. 79, pp. 2554-2558, 1982.

[10] S. Amari, "Characteristics of random nets of analog neuron-like elements," *IEEE Trans. Systems, Man and Cybernetics*, Vol. SMC-2, No. 5, pp.643-657, 1972.

[11] S. Grossberg and M. Cohen, "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks," *Trans. Syst., Man, Cybernetics*, Vol. SMC-13, pp. 815-826, 1983.

[12] J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," Proc. Natl. Acad. Sci. USA, May 1984.

[13] M. Vidyasagar, *Nonlinear System Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1978.

[14] J. Hopfield and D. Tank, "Neural computation of decisions in optimization problems," *Biol. Cybern.*, Vol. 52, pp. 141-152, 1985.

[15] D. Tank and J. Hopfield, "Simple 'neural' optimization networks: An A/D converter, signal decision circuit, and a liniear programming circuit," *IEEE Trans. Circuits Syst.*, Vol. 33, No. 5, pp. 533-541, 1986.

[16] Y.-T. Zhou, R. Chellappa, A. Vaid, K. Jenkins, "Image restoration using a neural network," *IEEE Trans. Acoust., Speech and Signal Processing*, Vol. 36, No. 7, pp. 1141-1151, 1988.

[17] A. Atiya, "Learning on a general network," in *Neural Information Processing Systems*, D. Anderson, *Ed.*, American Institute of Physics, New York, 1987.

[18] M. Hirsch, "Convergent activation dynamics in continuous time networks," *Neural Networks*, Vol. 2, No. 5, pp. 331-350, 1989.

[19] J. Franklin, *Matrix Theory*, Prentice-Hall, Englewood Cliffs, New Jersey, 1968.

[20] L. Almeida, "A learning rule for asynchronous perceptrons with feedback in a combinatorial environment," Proc. of the First Int. Annual Conf. on Neural Networks, San Diego, June 1987.

[21] F. Pineda, "Generalization of back-propagation to recurrent neural networks," Phys. Rev. Lett., vol. 59, no. 19, 9 Nov. 1987.

CHAPTER 2

A FEEDBACK ASSOCIATIVE MEMORY

2.1 INTRODUCTION

The brain has a remarkable ability to evoke stored memories using only partial or somewhat erroneous information. This has led a number of researchers to investigate how a collection of simple neuron-like elements can be used for real world applications such as content addressable memories, and pattern recognition.

The concept of the associative memory is the ability to reconstruct memorized items from incomplete or somewhat erroneous cues. There are two categories of associative memories: auto-associative memories, and hetero-associative memories. Consider the auto-associative memories. Let $\mathbf{y}(1), ..., \mathbf{y}(M)$ be a number of stored pattern vectors. The components of $\mathbf{y}(m)$ represent, for example, features extracted from the pattern. The function of the associative memory is to output a pattern vector $\mathbf{y}(m)$ when inputting a noisy or incomplete version of $\mathbf{y}(m)$. The other type of associative memory, the hetero-associative memory, is a more general concept. We have a number of key-response pairs $(\mathbf{c}(1), \mathbf{y}(1)), ..., (\mathbf{c}(M), \mathbf{y}(M))$. The memory outputs a response vector $\mathbf{y}(m)$ if a noisy or incomplete version of the key vector $\mathbf{c}(m)$ is given.

One of the well-known neural network associative memory models is the Hopfield discrete model [1] (see Section 1.2). It is a fully connected feedback network governed by asynchronous update. Hopfield showed that the model can store a number of binary vectors $\mathbf{y}(1), ..., \mathbf{y}(M)$ if we choose the weight matrix as

$$\mathbf{W} = \sum_{m=1}^{M} \left[\mathbf{y}(m)\mathbf{y}(m)^T - \mathbf{I}\right],$$

where $\mathbf{I}$ is the identity matrix. The basic idea of the model is that every memory vector is an equilibrium of the network. When presenting an erroneous memory vector, the state of the network evolves to the equilibrium representing the correct memory vector. Afterwards, many alternative techniques for storing binary vectors using also the discrete feedback net have been proposed, e.g. [2],[3],[4] and [5] (see also Michel and Farrell [6] for an overview). However, the problem of storing analog vectors, using Hopfield's continuous feedback network model of 1984 [7] (see Section 1.2), has received little attention in literature. By analog vectors we mean vectors whose components are real-valued. This problem is important because in a variety of applications of associative memories like pattern recognition and vector quantization, the patterns are originally in analog form and therefore one can save having the costly quantization step and therefore also save increasing the dimension of the vectors.

There are two main requirements for the associative memories. Every given memory should be an equilibrium of the network. Moreover, the equilibria corresponding to the memory vectors have to be asymptotically stable; i.e., they should be attractive (see definition in Section 1.3). Previous work on the design of associative memories using Hopfield's continuous model include the work by Farrell and Michel [8]. Their model can store binary vectors in the form of asymptotically stable equilibria. Pineda [9] developed an interesting method for the storage of analog vectors. In this method the network is designed using a learning procedure so that each given memory vector becomes an equilibrium of the network. It does not guarantee, though, the asymptotic stability of the equilibria. We propose here a method that solves the problem of storing analog vectors using the Hopfield continuous model. The network is designed so that each memory vector corresponds to an asymptotically stable equilibrium (see also a preliminary version of the method in Atiya and Abu-Mostafa [10]). Before describing the new method, we will study in the next section the Hopfield continuous model, and will examine its storage limitations.

2.2 THE NETWORK

Continuous-time neural networks can be modeled by the following differential equations (Hopfield [7]):

$$\frac{d\mathbf{u}}{dt} = -\mathbf{u} + \mathbf{W}\mathbf{f}(\mathbf{u}) + \theta \equiv \mathbf{h}(\mathbf{u}), \qquad \mathbf{x} = \mathbf{f}(\mathbf{u}). \tag{2.1}$$

where $\mathbf{u} = (u_1, ..., u_N)^T$ is the vector of neuron potentials, $\mathbf{x} = (x_1, ..., x_N)^T$ is the vector of firing rates, $\mathbf{W}$ is the weight matrix, the $(i,j)^{th}$ element being w_{ij}, $\theta = (\theta_1, ..., \theta_N)^T$ is the threshold vector, $\mathbf{f}(\mathbf{u})$ means the vector $\left(f(u_1), ..., f(u_N)^T\right)$, f is a sigmoid-shaped function, for example

$$f(u) = \tanh(u), \tag{2.2}$$

and N is the number of neurons.

If the weight matrix is symmetric and f is monotonically increasing, then the state always converges to an equilibrium (Grossberg and Cohen [11] and Hopfield [7]). The vectors to be stored are set as equilibria of the network. Giving a noisy version of any of the stored vectors as the initial state of the network, the network state has to eventually reach the equilibrium state corresponding to the correct vector. An important requirement is that these equilibria are asymptotically stable; otherwise, no attraction to the equilibria will take place. Unfortunately, there is a limitation on the set of vectors that can be stored, as given by the following theorem.

Theorem 2.1: If $\mathbf{W}$ is symmetric, $f(u)$ is monotonically increasing, $f(u)$ is strictly convex downwards for $u > 0$ and strictly convex upwards for $u < 0$, and $f(0) = 0$, then if $\mathbf{x}^*$ is a stored memory, no other memory $\mathbf{y}^*$ can be stored with

$$|y_i^*| \geq |x_i^*|, \qquad \text{sign}(y_i^*) = \text{sign}(x_i^*), \qquad \text{all } i \text{ such that } x_i^* \neq 0 \tag{2.3}$$

or with

$$|y_i^*| \leq |x_i^*|, \qquad \text{sign}(y_i^*) = \text{sign}(x_i^*), \qquad \text{all } i, \tag{2.4}$$

where

$$\text{sign}(x) = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{if } x = 0, \\ -1 & \text{if } x < 0. \end{cases}$$

Proof: Assume the contrary. First assume that there are two memories $\mathbf{x}^*$ and $\mathbf{y}^*$ satisfying (2.3). Both $\mathbf{x}^*$ and $\mathbf{y}^*$ are stored memories if and only if they are asymptotically stable equilibria of the system. We study the stability of the equilibrium $\mathbf{x}^*$ by linearizing (2.1) around the equilibrium (see Guckenheimer and Holmes [12]), to obtain

$$\frac{d\eta}{dt} = \mathbf{J}\eta,$$

where $\eta = \mathbf{u} - \mathbf{u}^*$, $\mathbf{x}^* = \mathbf{f}(\mathbf{u}^*)$ and $\mathbf{J}$ is the Jacobian matrix, the $(i,j)^{th}$ element being $\frac{\partial h_i}{\partial u_j}$, evaluated at $\mathbf{u}^*$, where h_i is the i^{th} element of $\mathbf{h}$ (of Eq. 2.1). It can be evaluated as

$$\mathbf{J} = -\mathbf{I} + \mathbf{W}\mathbf{\Lambda}(\mathbf{u}^*)$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{\Lambda}(\mathbf{u}^*)$ is the diagonal matrix whose i^{th} diagonal element is $f'(u_i^*)$. The equilibrium $\mathbf{u}^*$ in the non-linear system (2.1) can be asymptotically stable only if $\mathbf{J}$ has eigenvalues of negative or zero real values (see [12]). The eigenvalues of $\mathbf{J}$ are identical to those of $\left[-\mathbf{I} + \mathbf{\Lambda}^{1/2}(\mathbf{u}^*)\mathbf{W}\mathbf{\Lambda}^{1/2}(\mathbf{u}^*)\right]$ because if λ is an eigenvalue of $\mathbf{J}$, then

$$\det\left[-\mathbf{I} + \mathbf{W}\mathbf{\Lambda}(\mathbf{u}^*) - \lambda\mathbf{I}\right] = 0.$$

Upon multiplying both sides by $\det\left[\mathbf{\Lambda}^{1/2}(\mathbf{u}^*)\right]\det\left[\mathbf{\Lambda}^{-1/2}(\mathbf{u}^*)\right]$, we get

$$\det\left[-\mathbf{I} + \mathbf{\Lambda}^{1/2}(\mathbf{u}^*)\mathbf{W}\mathbf{\Lambda}^{1/2}(\mathbf{u}^*) - \lambda\mathbf{I}\right] = 0,$$

using the fact that f is monotone increasing and hence $\mathbf{\Lambda}(\mathbf{u}^*)$ is non-singular. The matrix $[-\mathbf{I} + \mathbf{\Lambda}^{1/2}(\mathbf{u}^*)\mathbf{W}\mathbf{\Lambda}^{1/2}(\mathbf{u}^*)]$ is negative semi-definite if and only if $[-\mathbf{\Lambda}^{-1}(\mathbf{u}^*) + \mathbf{W}]$ is negative semi-definite because for any vector $\mathbf{z}_1$, the quadratic form $\mathbf{z}_1^T[-\mathbf{\Lambda}^{-1}(\mathbf{u}^*) + \mathbf{W}]\mathbf{z}_1$ equals $\mathbf{z}_2^T[-\mathbf{I} + \mathbf{\Lambda}^{1/2}(\mathbf{u}^*)\mathbf{W}\mathbf{\Lambda}^{1/2}(\mathbf{u}^*)]\mathbf{z}_2$, where $\mathbf{z}_2 = \mathbf{\Lambda}^{-1/2}(\mathbf{u}^*)\mathbf{z}_1$. By the mean value theorem [13], there exists a point $\mathbf{x}$ on the straight line joining $\mathbf{x}^*$ and $\mathbf{y}^*$ ($\mathbf{x} \neq \mathbf{x}^*$, $\mathbf{x} \neq \mathbf{y}^*$) such that

$$(\mathbf{y}^*-\mathbf{x}^*)^T\mathbf{h}[\mathbf{f}^{-1}(\mathbf{y}^*)]-(\mathbf{y}^*-\mathbf{x}^*)^T\mathbf{h}[\mathbf{f}^{-1}(\mathbf{x}^*)] = \left[\frac{\partial(\mathbf{y}^*-\mathbf{x}^*)^T\mathbf{h}(\mathbf{f}^{-1}(\mathbf{x}))}{\partial\mathbf{x}}\right]^T(\mathbf{y}^*-\mathbf{x}^*),$$

(where $\mathbf{h}$ is as defined in (2.1)). But $\mathbf{h}[\mathbf{f}^{-1}(\mathbf{y}^*)] = \mathbf{h}[\mathbf{f}^{-1}(\mathbf{x}^*)] = 0$, since they are equilibria. We obtain

$$(\mathbf{y}^* - \mathbf{x}^*)^T[-\mathbf{\Lambda}^{-1}(\mathbf{u}) + \mathbf{W}](\mathbf{y}^* - \mathbf{x}^*) = 0, \tag{2.5}$$

where $\mathbf{u} = \mathbf{f}^{-1}(\mathbf{x})$. We observe that $\mathbf{x}$, being on the line joining $\mathbf{y}^*$ and $\mathbf{x}^*$, satisfies conditions

$$|x_i| \geq |x_i^*|, \qquad \text{sign}(x_i) = \text{sign}(x_i^*), \qquad \text{all } i \text{ such that } x_i^* \neq 0.$$

Because of the strict convexity assumption, $f'(u_i) < f'(u_i^*)$ when $|u_i| > |u_i^*|$. The left hand side of (2.5) can be written as

$$(\mathbf{y}^*-\mathbf{x}^*)^T[-\mathbf{\Lambda}^{-1}(\mathbf{u})+\mathbf{W}](\mathbf{y}^*-\mathbf{x}^*) = -\sum_i \frac{(y_i^* - x_i^*)^2}{f'(u_i)} + \sum_i\sum_j w_{ij}(y_i^*-x_i^*)(y_j^*-x_j^*).$$

Whenever y_i^* is strictly larger than x_i^*, then $f'(u_i) < f'(u_i^*)$, and hence the right hand side of the previous equation is strictly less than

$$-\sum_i \frac{(y_i^* - x_i^*)^2}{f'(u_i^*)} + \sum_i\sum_j w_{ij}(y_i^* - x_i^*)(y_j^* - x_j^*),$$

which in turn is less than or equal to zero because the Jacobian at $\mathbf{u}^*$ has negative or zero eigenvalues. Hence Equation (2.5) has no solution except when $\mathbf{x}^* = \mathbf{y}^*$, thus contradicting our assumption. The proof that there is no stable equilibrium $\mathbf{y}^*$ satisfying (2.4) follows by the argument that if this fact were not true, then by interchanging $\mathbf{x}^*$ and $\mathbf{y}^*$, condition (2.3) would be satisfied, which we proved to be impossible. ■

Figure (2.1) illustrates the limitations given by Theorem (2.1) for a two-dimensional case. Figure (2.1a) considers the case when one of the memories $\mathbf{x}$ has no zero component. The "forbidden regions," or regions where no other memory can be stored, are shown to be the two squares below and to the inside, and above and to the outside of the existing memory. Figure (2.1b) shows the case when one of the components of the existing memory is zero; then the forbidden region is now, in general, larger and extends over two quadrants. Figure (2.1c) illustrates an interesting corollary of the theorem, described as follows:

Corollary: If the origin is asymptotically stable then no other asymptotically stable, equilibrium can exist.

The convexity requirements on f of Theorem 2.1 are satisfied by the most commonly used forms of sigmoid functions, such as $f(u) = \tanh(u)$ and $f(u) = (2/\pi)\tan^{-1}(u)$. Even the absence of these requirements will change only the form but not necessarily the severity of the limitations.

The limitations given by Theorem 2.1 indicate some of the difficulties encountered in designing analog associative memories. There are sets of vectors that cannot be stored in the form of asymptotically stable equilibria. To solve this problem, we use an architecture consisting of both visible and hidden units. The outputs of the visible units correspond to the components of the stored vector. The purpose of the hidden units is to relieve the limitations on the visible components of the asymptotically stable equilibria. If the outputs of the hidden units are designed,

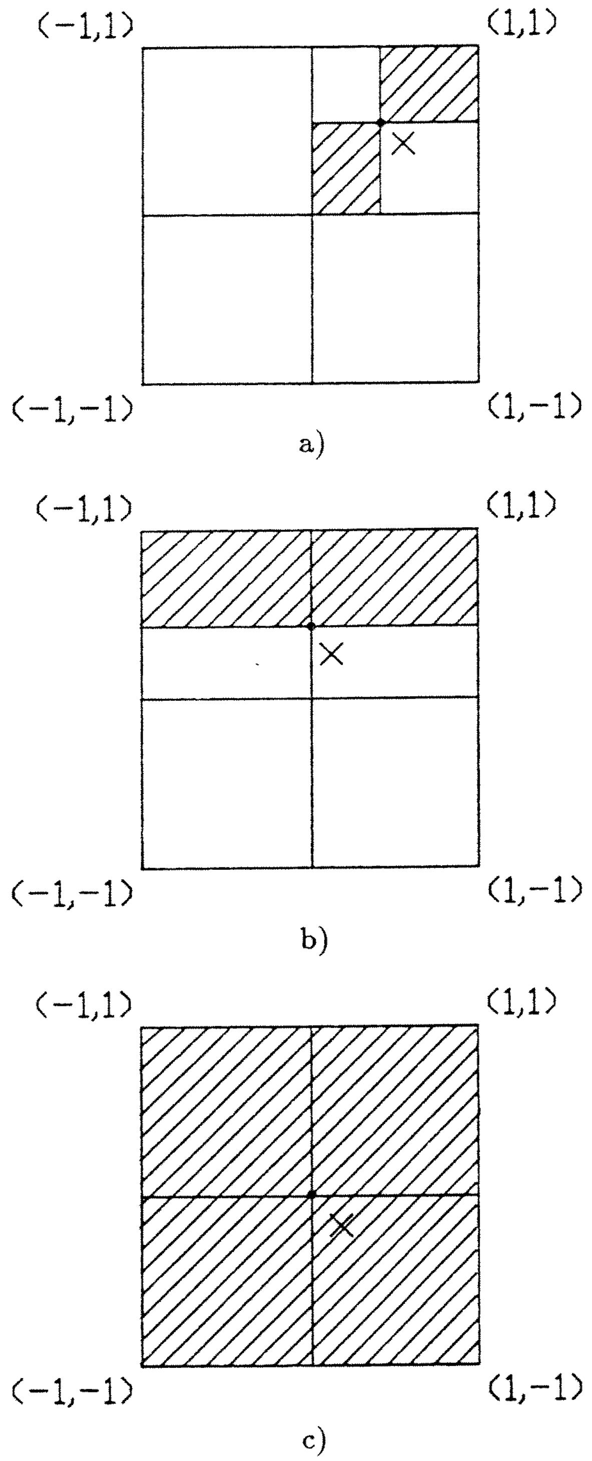

Figure 2.1: An illustration of the limitations of the Hopfield continuous model on the set of memory vectors that can be stored. Consider a two-dimensional case. x represents one of the stored vectors. The hatched regions, or the "forbidden regions," represent the regions where no other memory can be stored.

for example, to have some components of a different sign for every different stored vector, then Theorem 2.1 does not provide any limitations on the freedom of choosing the sets of memory vectors to be stored (of course, under the assumption of having enough hidden units). We will not restrict ourselves to a symmetric weight matrix. Our proposed model is as follows. We have a number of hidden layers and one visible layer, arranged in a circular fashion, with connections running in one direction from one layer to the next, as shown in Figure (2.2). No connections exist within each of the layers. The advantage of such an architecture is that it lends itself to a simple design procedure, as we will see in the next section. It is worth noting that such an architecture has been considered before, for various purposes. In Chapter 4 we have analyzed oscillations and phase locking phenomena for such an architecture. Kosko [14],[15] considered the special case of a two-layer network and proposed a model for hetero-associative memory for binary vectors called the Balanced Associative Memory (BAM), using discrete neurons.

Let $\mathbf{x}^{(l)}$ be the output vector of layer l. Then, our model is governed by the following set of differential equations,

$$\frac{d\mathbf{u}^{(l)}}{dt} = -\mathbf{u}^{(l)} + \mathbf{W}^{(l,l-1)}\mathbf{f}(\mathbf{u}^{(l-1)}) + \theta^{(l)}, \qquad \mathbf{x}^{(l)} = \mathbf{f}(\mathbf{u}^{(l)}), \tag{2.6}$$

where $\mathbf{W}^{(l,l-1)}$ is the $N_l \times N_{l-1}$ matrix of weights from layer $l-1$ to layer l ($\mathbf{W}^{(1,0)} \equiv \mathbf{W}^{(1,L)}$, $\mathbf{u}^{(0)} \equiv \mathbf{u}^{(L)}$), $\theta^{(l)}$ is the threshold vector for layer l, f is a sigmoid function in the range from -1 to 1, and L is the number of layers. The layers from 1 to $L-1$ represent the hidden layers, whereas layer L represents the visible layer.

We would like each memory vector to correspond to the visible layer component of an asymptotically stable equilibrium of the network. We have given a motivation for the choice of the architecture by saying that for this choice Theorem 2.1 does not provide any limitations on the sets of memories that can be stored. However, it has to be shown that no limitations of any other kind whatsoever will be

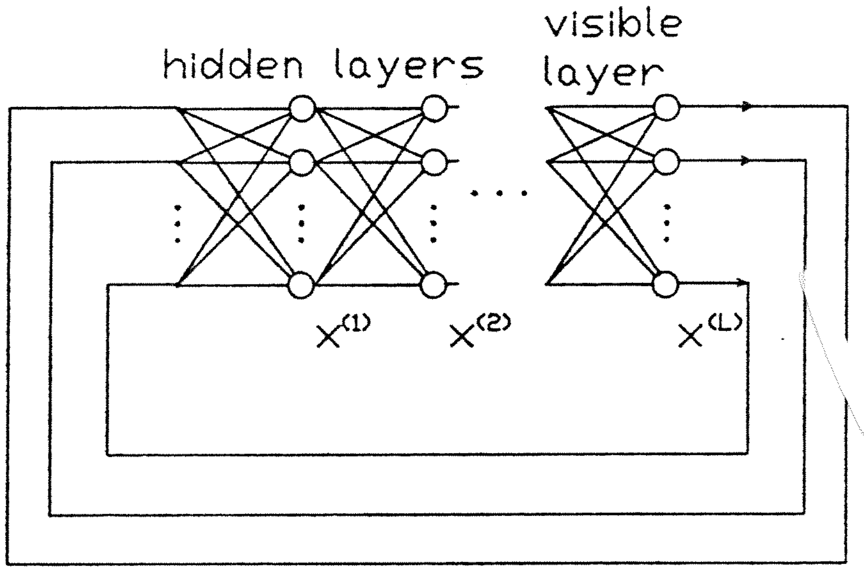

Figure 2.2: The proposed architecture for the new associative memory

encountered. The following theorem gives conditions for the stability of equilibria in the considered architecture.

Theorem 2.2: An equilibrium point

$$\mathbf{u}^* = \begin{pmatrix} \mathbf{u}^{*(1)} \\ \vdots \\ \mathbf{u}^{*(L)} \end{pmatrix}$$

satisfying

$$\sum_{j=1}^{N_l} |a_{ij}| f'(u_j^{*(l)}) < 1, \qquad i = 1, ..., N_l \tag{2.7}$$

for some l is asymptotically stable, where a_{ij} is the $(i,j)^{th}$ element of a matrix $\mathbf{A}$, given by

$$\mathbf{A} = \mathbf{W}^{(l,l-1)}\mathbf{\Lambda}(\mathbf{u}^{*(l-1)}) \ldots \mathbf{W}^{(2,1)}\mathbf{\Lambda}(\mathbf{u}^{*(1)})\mathbf{W}^{(1,L)}\mathbf{\Lambda}(\mathbf{u}^{*(L)}) \ldots \mathbf{\Lambda}(\mathbf{u}^{*(l+1)})\mathbf{W}^{(l+1,l)}.$$

Proof: We linearize (2.6) around the equilibrium $\mathbf{u}^*$. We get

$$\frac{d\eta}{dt} = \mathbf{J}\eta,$$

where $\eta = \mathbf{u} - \mathbf{u}^*$ and $\mathbf{J}$ is the Jacobian matrix, evaluated as

$$\mathbf{J} = -\mathbf{I} + \begin{pmatrix} \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{W}^{(1,L)}\mathbf{\Lambda}(\mathbf{u}^{*(L)}) \\ \mathbf{W}^{(2,1)}\mathbf{\Lambda}(\mathbf{u}^{*(1)}) & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{W}^{(L,L-1)}\mathbf{\Lambda}(\mathbf{u}^{*(L-1)}) & \mathbf{0} \end{pmatrix}$$

Let λ be an eigenvalue of $\mathbf{J}$, associated with eigenvector $\mathbf{v} = \left(\mathbf{v}_1^T, ..., \mathbf{v}_L^T\right)^T$, where $\mathbf{v}_l$ represents the component corresponding to layer l. Then,

$$\mathbf{W}^{(1,L)}\mathbf{\Lambda}(\mathbf{u}^{*(L)})\mathbf{v}_L = (\lambda + 1)\mathbf{v}_1,$$

$$\mathbf{W}^{(2,1)}\mathbf{\Lambda}(\mathbf{u}^{*(1)})\mathbf{v}_1 = (\lambda + 1)\mathbf{v}_2,$$

$$\vdots$$

$$\mathbf{W}^{(L,L-1)}\mathbf{\Lambda}(\mathbf{u}^{*(L-1)})\mathbf{v}_{L-1} = (\lambda + 1)\mathbf{v}_L,$$

from which we obtain

$$\mathbf{A}\mathbf{\Lambda}(\mathbf{u}^{*(l)})\mathbf{v}_l = (\lambda + 1)^L\mathbf{v}_l.$$

By Gershgorin's Theorem [16], any eigenvalue β of $\mathbf{A}\mathbf{\Lambda}(\mathbf{u}^{*(l)})$ satisfies one of the following N_l inequalities:

$$|\beta| \leq \sum_{j=1}^{N_l} |a_{ij}| f'(u_j^{*(l)}), \qquad i = 1, ..., N_l.$$

Since $\lambda = -1 + \beta^{1/L}$, then if Inequalities (2.7) are satisfied, λ will have a negative real part. Thus, the equilibrium $\mathbf{u}^*$ of the system (2.6) is asymptotically stable. ∎

Thus, if the neurons of one of the hidden layers are driven far enough into the saturation region, then the corresponding equilibrium will be stable because then, $f'(u_i^*)$ will be very small, causing Inequalities (2.7) to be satisfied. This provides a motivation for the learning algorithm, which is described in the next section.

2.3 TRAINING THE NETWORK

Let $\mathbf{y}(m)$, $m = 1, ..., M$ be the vectors to be stored. Each $\mathbf{y}(m)$ corresponds to the visible layer component of an equilibrium. Let the equilibrium be denoted by $\mathbf{x}(m)$, where

$$\mathbf{x}(m) = \begin{pmatrix} \mathbf{x}^{(1)}(m) \\ \vdots \\ \mathbf{x}^{(L)}(m) \end{pmatrix},$$

where clearly $\mathbf{x}^{(L)}(m) = \mathbf{y}(m)$. We choose arbitrarily one of the hidden layers, say layer K, and design the network such that the equilibrium values of that layer $x_i^{(K)}$, $i = 1, ..., N_K$ are very close to 1 or -1. This means that the neurons in that layer are saturated, and we will therefore call this layer the saturating layer. Assuming that the outputs of the neurons in the saturating layer are close enough to 1 or -1, then Theorem 2.2 guarantees the asymptotic stability of the equilibrium. Enforcing the saturation of the units in the saturating layer is quite essential, since simulations show that when ignoring stability requirements, the equilibrium is probably unstable. Since we have an asymmetric network, there is nothing to rule out the existence of limit cycles. Also, we can have spurious equilibria, i.e. equilibria that correspond to no memory vector. However, if these unwanted limit cycles and spurious equilibria occur, then they would be far away from the memory vectors because each memory vector is guaranteed to have a basin of attraction around it.

The design of the memory is as follows. For each memory vector $\mathbf{y}(m)$ we choose an arbitrary target vector $\mathbf{z}(m)$ for the saturating layer, of the form $(\pm 1, ..., \pm 1)^T$. A simple way is to generate the $z_i(m)$'s randomly. We will call the $\mathbf{z}(m)$'s the *saturating layer vectors.*

Let us write the equations for the equilibrium of (2.6). Equating the right hand side to zero leads to:

$$\mathbf{x}^{(l)}(m) = \mathbf{f}(\mathbf{W}^{(l,l-1)}\mathbf{x}^{(l-1)}(m) + \theta^{(l)}), \qquad l = 1, ..., L. \tag{2.8}$$

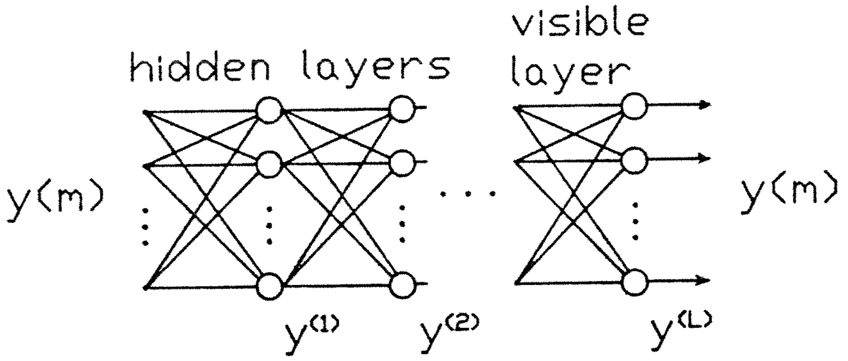

Figure 2.3: The equivalent feedforward network

The problem can be reduced to training an equivalent feedforward network (see Fig. (2.3)) to map input vector $\mathbf{y}(m)$ to output vector $\mathbf{y}(m)$ for $m = 1, ..., M$. This becomes clear if we compare the equation of the equilibrium for our feedback network (Eq. (2.8)) with the equation for the output of the feedforward network:

$$\mathbf{y}^{(l)}(m) = \mathbf{f}(\mathbf{W}^{(l,l-1)}\mathbf{y}^{(l-1)}(m) + \theta^{(l)}), \qquad l = 1, ..., L,$$

where the input $\mathbf{y}^{(0)}(m)$ is $\mathbf{y}(m)$ ($\mathbf{y}^{(l)}(m)$ is the output of layer l for the feedforward network).

We define the two error functions E_1 and E_2,

$$E_1 = \sum_{m=1}^{M} E_1(m), \qquad E_1(m) = \|\mathbf{y}^{(K)}(m) - \mathbf{z}(m)\|^2,$$

$$E_2 = \sum_{m=1}^{M} E_2(m), \qquad E_2(m) = \|\mathbf{y}^{(L)}(m) - \mathbf{y}(m)\|^2.$$

Training is performed by applying a backpropagation type algorithm twice (see the description of the backpropagation algorithm in Section 1.1). First we train the layers 1 to K to map the input vector $\mathbf{y}(m)$ to the saturating layer vector $\mathbf{z}(m)$, by minimizing E_1 for the feedforward network. Then, we train layers $K + 1$ to L to map the output of the saturating layer K to the output vector $\mathbf{y}(m)$, by minimizing E_2.

We note that we use a slightly modified version of the backpropagation algorithm for training the layers 1 to K. A standard backpropagation algorithm will have a very slow convergence because the targets are 1 or -1. Therefore, in our method we used a stopping criterion as follows. If the output signs match the target signs and the outputs are at least some constant β in magnitude (a typical value $\beta = 0.8$), then we stop iterating the weights. We then multiply the weights and the thresholds of the saturating layer by a big positive constant in order to drive the outputs of the saturating layer as close as possible to 1 or -1. Let ϵ_1 and ϵ_2 be two small, positive constants (typical values: $\epsilon_1 = 0.01$ and $\epsilon_2 = 10^{-5}$), and let ρ denote the step size. Our algorithm is as follows:

1) Generate the initial setting of the weight matrices $\mathbf{W}^{(l,l-1)}(0)$ and $\theta^{(l)}(0)$ randomly, $l = 1, ..., L$.

2) Apply the steepest descent update (the delta rule, refer to Section 1.1) for E_1 for the first K layers,

$$\mathbf{W}^{(l,l-1)}(n) = \mathbf{W}^{(l,l-1)}(n-1) - \rho \frac{\partial E_1}{\partial \mathbf{W}^{(l,l-1)}}, \qquad l = 1, ..., K,$$

$$\theta^{(l)}(n) = \theta^{(l)}(n-1) - \rho \frac{\partial E_1}{\partial \theta^{(l)}}, \qquad l = 1, ..., K,$$

until $\text{sign}\left[y_i^{(K)}(m)\right] = \text{sign}\left[z_i(m)\right]$, and $|y_i^{(K)}(m)| \geq \beta$ for all i and m.

3) Set

$$\mathbf{W}^{(K,K-1)} = c\mathbf{W}^{(K,K-1)}(n),$$

$$\theta^{(K)} = c\theta^{(K)}(n),$$

where c is a positive constant such that $|y_i^{(K)}(m)| \geq 1 - \epsilon_1$ for all i and m.

4) Apply the steepest descent update for E_2 for layers $K+1$ to L,

$$\mathbf{W}^{(l,l-1)}(n) = \mathbf{W}^{(l,l-1)}(n-1) - \rho\frac{\partial E_2}{\partial \mathbf{W}^{(l,l-1)}}, \qquad l = K+1, ..., L,$$

$$\theta^{(l)}(n) = \theta^{(l)}(n-1) - \rho\frac{\partial E_2}{\partial \theta^{(l)}}, \qquad l = K+1, ..., L,$$

until $E_2 < \epsilon_2$.

Remarks:

a) It is important to note that for training layers $K+1$ to L, the input vector to layer $K+1$ is, of course, taken to be the output vector of layer K, not the target vector $\mathbf{z}(m)$ of the saturating layer, because they are slightly different.

b) After learning has been completed, the stability of the equilibria has to be checked, by simulating the set of differential equations, or else by checking Ineq. (2.7). If some of the equilibria are unstable, then this means that we have not chosen a small enough ϵ_1. In our simulations we have never encountered an unstable equilibrium when $\epsilon_1 = 0.02$.

c) We have applied also the other variant of the backpropagation algorithm, where we update the weights after presenting each training input instead of after cycling through all the training inputs (see Section 1.1). We observed that it generally converges faster.

d) Let $\mathbf{W}$ and θ represent respectively the weight matrix and threshold vector at which Equations (2.8) are satisfied. Practically, the learning algorithm might result in an output of the feedforward network slightly different from $\mathbf{y}(m)$, when inputting $\mathbf{y}(m)$, and a parameter set $\mathbf{W}'$ and θ' slightly different from $\mathbf{W}$ and θ, respectively. We assert that this produces only a slight displacement of the equilibrium of the feedback network. This is because the equilibrium is stable and hence the corresponding Jacobian is non-singular (for example, there will be no bifurcations at which the equilibrium might disappear).

e) A very important remark: Once learning is completed, retrieving a memory vector $\mathbf{y}(m)$ using a noisy version $\mathbf{y}'(m)$ is performed as follows. The initial condition for the visible layer is set as $\mathbf{y}'(m)$. The initial conditions for the remaining layers are set as

$$\mathbf{x}^{(l)} = \mathbf{f}(\mathbf{W}^{(l,l-1)}\mathbf{x}^{(l-1)} + \theta^{(l)}), \qquad l = 1, ..., L-1,$$

where $\mathbf{x}^{(0)} = \mathbf{x}^{(L)} \equiv \mathbf{y}'(m)$. A practical way to load the initial conditions is by clamping $\mathbf{y}'(m)$ shortly onto the visible layer and allowing the network to evolve according to its equations of motion (2.6), while the connections from the last hidden layer to the visible layer are temporarily switched off.

f) The presented training method applies also to the discrete-time continuous-output update formulation, i.e. the system given by

$$\mathbf{x}^{(l)}(t+1) = \mathbf{f}(\mathbf{W}^{(l,l-1)}\mathbf{x}^{(l-1)}(t) + \theta^{(l)}), \qquad l = 1, ..., L.$$

The equilibria for such a system are equivalent to those of the continuous-time system (2.6). Also, we can show using a theorem analogous to Theorem 2.2 that if one of the hidden layers is heavily saturated, then the corresponding equilibrium is asymptotically stable. This system has yet to be tested by numerical simulations, to verify its global stability properties.

2.4 HETERO-ASSOCIATIVE MEMORY

Any auto-associative memory can be easily extended into a hetero-associative memory by applying an appropriate linear transformation at the output. We present a more direct way. Let $(\mathbf{c}(1), \mathbf{y}(1)), ..., (\mathbf{c}(M), \mathbf{y}(M))$ be the pairs of stored associations, and $\mathbf{z}(1), ..., \mathbf{z}(M)$ be the corresponding saturating layer vectors. We have at least 3 layers in our loop architecture: one saturating layer K_1, and two layers K_2 and L, which upon convergence are to carry $\mathbf{c}(m)$ and $\mathbf{y}(m)$, respectively. In a way, this is a multi-layer analog extension of Kosko's two-layer hetero-associative BAM [14],[15]. Consider also an equivalent feedforward network as in the previous section. Let

$$E_1 = \sum_{m=1}^{M} E_1(m), \qquad E_1(m) = \|\mathbf{y}^{(K_1)}(m) - \mathbf{z}(m)\|^2,$$

$$E_2 = \sum_{m=1}^{M} E_2(m), \qquad E_2(m) = \|\mathbf{y}^{(K_2)}(m) - \mathbf{c}(m)\|^2,$$

$$E_3 = \sum_{m=1}^{M} E_3(m), \qquad E_3(m) = \|\mathbf{y}^{(L)}(m) - \mathbf{y}(m)\|^2.$$

Learning is performed according to the following straightforward extension of the learning algorithm of the previous section:

1) Generate the initial setting of the weight matrices $\mathbf{W}^{(l,l-1)}(0)$ and $\theta^{(l)}(0)$ randomly, $l = 1, ..., L$.

2) Apply the steepest descent update for E_1 for the first K_1 layers,

$$\mathbf{W}^{(l,l-1)}(n) = \mathbf{W}^{(l,l-1)}(n-1) - \rho \frac{\partial E_1}{\partial \mathbf{W}^{(l,l-1)}}, \qquad l = 1, ..., K_1,$$

$$\theta^{(l)}(n) = \theta^{(l)}(n-1) - \rho \frac{\partial E_1}{\partial \theta^{(l)}}, \qquad l = 1, ..., K_1,$$

until $\text{sign}[y_i^{(K_1)}(m)] = \text{sign}[z_i(m)]$, and $|y_i^{(K_1)}(m)| \geq \beta$ for all i and m.

3) Set

$$\mathbf{W}^{(K_1,K_1-1)} = c\mathbf{W}^{(K_1,K_1-1)}(n),$$

$$\theta^{(K_1)} = c\theta^{(K_1)}(n),$$

where c is a positive constant such that $|y_i^{(K_1)}(m)| \geq 1 - \epsilon_1$ for all i and m.

4) Apply the steepest descent update for E_2 for layers K_1+1 to K_2,

$$\mathbf{W}^{(l,l-1)}(n) = \mathbf{W}^{(l,l-1)}(n-1) - \rho\frac{\partial E_2}{\partial \mathbf{W}^{(l,l-1)}}, \qquad l = K_1+1,...,K_2,$$

$$\theta^{(l)}(n) = \theta^{(l)}(n-1) - \rho\frac{\partial E_2}{\partial \theta^{(l)}}, \qquad l = K_1+1,...,K_2,$$

until $E_2 < \epsilon_2$.

5) Apply the steepest descent update for E_3 for layers K_2+1 to L,

$$\mathbf{W}^{(l,l-1)}(n) = \mathbf{W}^{(l,l-1)}(n-1) - \rho\frac{\partial E_3}{\partial \mathbf{W}^{(l,l-1)}}, \qquad l = K_2+1,...,L,$$

$$\theta^{(l)}(n) = \theta^{(l)}(n-1) - \rho\frac{\partial E_3}{\partial \theta^{(l)}}, \qquad l = K_2+1,...,L,$$

until $E_3 < \epsilon_2$.

2.5 A BINARY ASSOCIATIVE MEMORY

In case of a binary associative memory, a method simpler than the one presented in Section 2.3 is even possible. Let the memory vectors $\mathbf{y}(1),...,\mathbf{y}(M)$ be of the form $(\pm1,...,\pm1)^T$. At an equilibrium corresponding to some memory vector $\mathbf{y}(m)$, the visible layer is heavily saturated, and therefore by Theorem 2.2 the equilibrium is stable, even if we do not have a saturating hidden layer. A saturating hidden layer is therefore not needed, and the problem reduces to applying a backpropagation algorithm only once on the feedforward network of Fig. (2.3). The set of input vectors are $\mathbf{y}(1),...,\mathbf{y}(M)$, and the corresponding target output vectors at the last layer L are again $\mathbf{y}(1),...,\mathbf{y}(M)$. Let

$$E_B = \sum_{m=1}^{M} \|\mathbf{y}^{(L)}(m) - \mathbf{y}(m)\|^2.$$

The algorithm is as follows:

1) Generate the initial setting of the weight matrices $\mathbf{W}^{(l,l-1)}(0)$ and $\theta^{(l)}(0)$ randomly, $l = 1,...,L$.

2) Apply the steepest descent update for E_B for all L layers,

$$\mathbf{W}^{(l,l-1)}(n) = \mathbf{W}^{(l,l-1)}(n-1) - \rho \frac{\partial E_B}{\partial \mathbf{W}^{(l,l-1)}}, \qquad l = 1, ..., L,$$

$$\theta^{(l)}(n) = \theta^{(l)}(n-1) - \rho \frac{\partial E_B}{\partial \theta^{(l)}}, \qquad l = 1, ..., L,$$

until sign$\left[y_i^{(L)}(m)\right]$ =sign$\left[z_i(m)\right]$, and $|y_i^{(L)}(m)| \geq \beta$ for all i and m.

3) Set

$$\mathbf{W}^{(L,L-1)} = c\mathbf{W}^{(L,L-1)}(n),$$

$$\theta^{(L)} = c\theta^{(L)}(n),$$

where c is a positive constant such that $|y_i^{(L)}(m)| \geq 1 - \epsilon_1$ for all i and m.

2.6 THE CAPACITY OF THE NETWORK

On the Capacity of Continuous-time Networks

For the Hopfield network, described by Eq. (2.1), it was shown by Guez *et al.* [17] that there are at most 3^N equilibria. We can show that we can obtain 2^N asymptotically stable equilibria by considering a diagonal weight matrix with sufficiently large entries. For the architecture considered, it is possible to show that the number of asymtotically stable equilibria we can achieve is at least 2 to the power the number of neurons in the smallest layer. However, the issue here is not so much how many equilibria we can have, as much as how programmable their locations are. Consider a general fully connected continuous Hopfield network (Eq. 2.1) with I visible neurons and J hidden neurons. The network should work as an associative memory, in the sense that each memory vector $\mathbf{y}(m)$ corresponds to an equilibrium. Let M be the number of memory vectors, and N be the total number of neurons (i.e. $N = I + J$). Then we have MI equations for the equilibria, while there are $(I+J)(I+J+1)$ degrees of freedom, corresponding to the weights and the thresholds. Thus the capacity is at most

$$\frac{(I+J)(I+J+1)}{I}. \tag{2.9}$$

The set of vectors that can be stored if M is greater than (2.9) represents the range of the mapping, which maps $(\mathbf{W}, \theta)$ to $(\mathbf{y}(1), ..., \mathbf{y}(M))$ if $\mathbf{y}(1), ..., \mathbf{y}(M)$ are stable equilibria of the network, and therefore it is a number of surfaces of a lower dimension embedded in the space of all possible sets of M memory vectors. Therefore, if M is greater than (2.9), then the fraction of all possible sets of vectors that can be stored is virtually a set of measure zero.

Consider a fully connected network with no hidden neurons. Apart from the constraint shown in Theorem 2.1 on the set of vectors that can be stored, it follows from the previous argument that only a zero-fraction of the set of all possible M vectors can be stored, if $M > N+1$. Thus, there is an abrupt change when M exceeds $N+1$. In case of a symmetric matrix, the abrupt change occurs at $(N+3)/2$.

The Capacity of the Proposed Model

An expression for the capacity is difficult to obtain for the general multi-layer network. The two-layer case is of special interest because of its simplicity and the adequacy of its capabilities. Let N_1 and N_2 be the numbers of neurons in the hidden and the visible layers, respectively. Using an argument similar to the one presented in the previous subsection, we can conclude that the capacity cannot exceed $(2N_1N_2 + N_1 + N_2)/N_2$, which is approximately $2N_1$ for large N_1 and N_2. Of more interest is to obtain a lower bound on the capacity, since we are interested more in what the network can do rather than in what it cannot do. The following theorem gives a lower bound. First, define the *augmented memory vectors* and the *augmented saturating layer vectors* as, respectively, $\left[\mathbf{y}(1)^T|1\right]^T, ..., \left[\mathbf{y}(M)^T|1\right]^T$ and

$\left[\mathbf{z}(1)^T|1\right]^T, ..., \left[\mathbf{z}(M)^T|1\right]^T$.

Theorem 2.3: A two-layer network is guaranteed to store any set of $N_1 + 1$ vectors.

Proof: Sufficient conditions for the storage of the memory vectors are:

$$\left[\mathbf{W}^{(2,1)}|\theta^{(2)}\right]\mathbf{Z} = \mathbf{V}, \tag{2.10}$$

$$\left[\mathbf{W}^{(1,2)}|\theta^{(1)}\right]\mathbf{Y} = \mathbf{F}, \tag{2.11}$$

and

$$\sum_{j=1}^{N_1} |a_{ij}| f'(u_j(m)) < 1, \qquad i = 1, ..., N_1, \tag{2.12}$$

where $\mathbf{Z}$ is the matrix whose columns are the augmented saturating layer vectors, $\mathbf{Y}$ is the matrix whose columns are the augmented memory vectors, $\mathbf{V}$ and $\mathbf{F}$ are the matrices given by

$$\mathbf{V} = \begin{pmatrix} f^{-1}\left[y_1(1)\right] & \dots & f^{-1}\left[y_1(M)\right] \\ \vdots & \vdots & \vdots \\ f^{-1}\left[y_{N_2}(1)\right] & \dots & f^{-1}\left[y_{N_2}(M)\right] \end{pmatrix},$$

$$\mathbf{F} = \begin{pmatrix} f^{-1}\left[z_1(1)\right] & \dots & f^{-1}\left[z_1(M)\right] \\ \vdots & \vdots & \vdots \\ f^{-1}\left[z_{N_1}(1)\right] & \dots & f^{-1}\left[z_{N_1}(M)\right] \end{pmatrix},$$

$u_j(m) = f^{-1}(z_j(m))$, and a_{ij} is the $(i,j)^{th}$ element of the matrix $\mathbf{A}$, defined as $\mathbf{A} = \mathbf{W}^{(1,2)}\mathbf{\Lambda}\left[\mathbf{f}^{-1}(\mathbf{y}(m))\right]\mathbf{W}^{(2,1)}$. Equations (2.10) and (2.11) guarantee that the memory vectors are equilibria, whereas Inequalities (2.12) ensure their stability, according to Theorem 2.2.

Let $\mathbf{Z}$ be given by the following $(N_1 + 1) \times M$ matrix:

$$\mathbf{Z} = \begin{pmatrix} 1 & -1 & -1 & \dots & -1 \\ 1 & 1 & -1 & \dots & -1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & 1 & \dots & -1 \\ 1 & 1 & 1 & \dots & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & 1 & \dots & 1 \end{pmatrix}.$$

Equation (2.10) represents N_2 sets of linear equations in the weights and the thresholds. A solution for $\left[\mathbf{W}^{(2,1)}|\theta^{(2)}\right]$ exists if $M \le N_1 + 1$, since the columns of $\mathbf{Z}$ are linearly independent. Consider Equation (2.11). It is enough to prove that there is a matrix $\mathbf{W}$ and a threshold vector θ satisfying

$$\text{sign}\left[\mathbf{W}\mathbf{y}(m) + \theta\right]_i = \text{sign}\left[z_i(m)\right] \qquad \text{all } i, m, \tag{2.13}$$

because one can then take $\mathbf{W}^{(1,2)} = \alpha\mathbf{W}$, and $\theta^{(1)} = \alpha\theta$, where α is big enough to drive the outputs of the hidden units far into the saturation region, i.e., very close to -1 or 1. It is possible to find a $\mathbf{W}$ and θ to satisfy Equation (2.13), as follows. One can take $\mathbf{W} = (\mathbf{w}|...|\mathbf{w})^T$, where $\mathbf{w}$ is a column vector satisfying

$$\mathbf{w}^T\mathbf{y}(i) \neq \mathbf{w}^T\mathbf{y}(j) \qquad \text{for } i \neq j. \tag{2.14}$$

Such a $\mathbf{w}$ exists for any set of distinct vectors $\mathbf{y}(1), ..., \mathbf{y}(M)$, because otherwise $\mathbf{w}$ can be perturbed to restore (2.14). Rename the indices of the memories such that

$$\mathbf{w}^T\mathbf{y}(1) > \mathbf{w}^T\mathbf{y}(2) > \ldots > \mathbf{w}^T\mathbf{y}(M).$$

We choose the θ_i's, such that

$$\mathbf{w}^T\mathbf{y}(1) > -\theta_1 > \mathbf{w}^T\mathbf{y}(2) > -\theta_2 > \ldots > \mathbf{w}^T\mathbf{y}(M) > -\theta_M > \ldots > \theta_{N_1}.$$

We can see that this choice satisfies (2.13). Now we show that taking α large enough will satisfy Ineq.(2.12). We take the limit as $\alpha \to \infty$. We get

$$\lim_{\alpha\to\infty}\sum_{j=1}^{N_1}|a_{ij}|\alpha f'\left(\alpha u_j(m)\right) = \lim_{\alpha\to\infty}\sum_{j=1}^{N_1}|a_{ij}|\alpha\frac{4e^{-2\alpha u_j(m)}}{(1+e^{-2\alpha u_j(m)})^2} = 0 < 1,$$

where we used the sigmoid function defined in (2.2). This completes the proof that $N_1 + 1$ memory vectors correspond to asymptotically stable equilibrium points. ∎

Thus, the storage capacity equals 1+number of hidden neurons. For the network to achieve this capacity, a particular choice of the saturating layer vectors has to be taken. In our algorithm the saturating layer vectors are specified by the user, as generating them randomly, for example, realizes the distributiveness of the memory as well as results in good retrieval accuracy. The question is whether for any choice of saturating layer vectors the capacity will remain the same as $N_1 + 1$. The answer turns out to be negative, and the capacity can be lower in this case, as will be shown in the next theorem.

Theorem 2.4: If the augmented memory vectors are linearly independent, and the augmented saturating layer vectors are linearly independent, then a two-layer network is guaranteed to store $\min(N_1, N_2) + 1$ vectors.

Proof: The memory vectors correspond to the stable equilibria if conditions (2.10)–(2.12) are satisfied. Eq. (2.10) has a solution for $\mathbf{W}^{(2,1)}$, since $M \leq N_1 + 1$ and the columns of $\mathbf{Z}$ are linearly independent by hypothesis. Also, Eq. (2.11) has a solution for $\mathbf{W}^{(1,2)}$ because $M \leq N_2 + 1$ and the columns of $\mathbf{Y}$ are linearly independent, again by the hypothesis of the theorem. Inequalities (2.12) are satisfied by choosing the magnitudes of the elements of $\mathbf{F}$ large enough and using an argument similar to that in the proof of Theorem 2.3. ∎

If $M > \min(N_1, N_2) + 1$, then Eq. (2.10) and (2.11) will constitute more equations than unknowns. Hence, with a random choice of the saturating layer vectors, there is a small chance for being able to store a given set of vectors. We must caution, however, that the previous theorem gives only the number of vectors that can be stored when the saturating layer vectors are given. It does not guarantee their storage when using the learning algorithm presented in Section 2.3. This is because it is possible to get stuck in a local minimum of the error function. Although there is a simple way to satisfy the conditions (2.10)–(2.12), sufficient for the storage of the memory vectors, by simply solving the linear Equations (2.10) and (2.11), we favor using the learning method for the following reason. If we are not at full capacity, then Equations (2.10) and (2.11) do not have a unique solution. Specifying a solution arbitrarily by, for example, randomly generating the values of the extra degrees of freedom has led to results quite inferior to those of the learning method.

2.7 IMPLEMENTATION

The theoretical results presented in the last section indicate some of the capabilities of the new associative memory. However, some of the issues remain yet unresolved by the theoretical analysis, such as the extent of the existence of limit cycles and spurious equilibria, and the shapes of the basins of attraction. We have therefore resorted to numerical simulations. In the numerous experiments we have performed we have never encountered limit cycles. Spurious equilibria exist, but their number is high only when we are close to full capacity. Concerning the basins of attraction, they are required to have balanced shapes (for example, no memory vector has a too small or a too large basin of attraction). We have tested this by performing the following experiment. We have a two-layer network with two hidden neurons and two visible neurons. The two vectors $(-0.5, -0.5)^T$ and $(0.75, 0.25)^T$ are stored. Figure (2.4) shows the resulting basins of attraction.

To test the error correcting capability of the model, we have performed the following experiment. A two-layer network is considered with 10 neurons in the hidden layer and 10 neurons in the visible layer. Five randomly generated memory vectors are considered, with components ranging from -0.8 to 0.8 (if we have components close to 1 or -1, then convergence in learning can get very slow because we are close to the saturation region). Thus, we are operating at about half capacity. After learning, the model is tested by giving memory vectors corrupted by noise (100 vectors for a given variance). Fig. (2.5) shows the error fraction versus the signal-to-noise ratio. The model is also tested for pattern completion ability. Com-

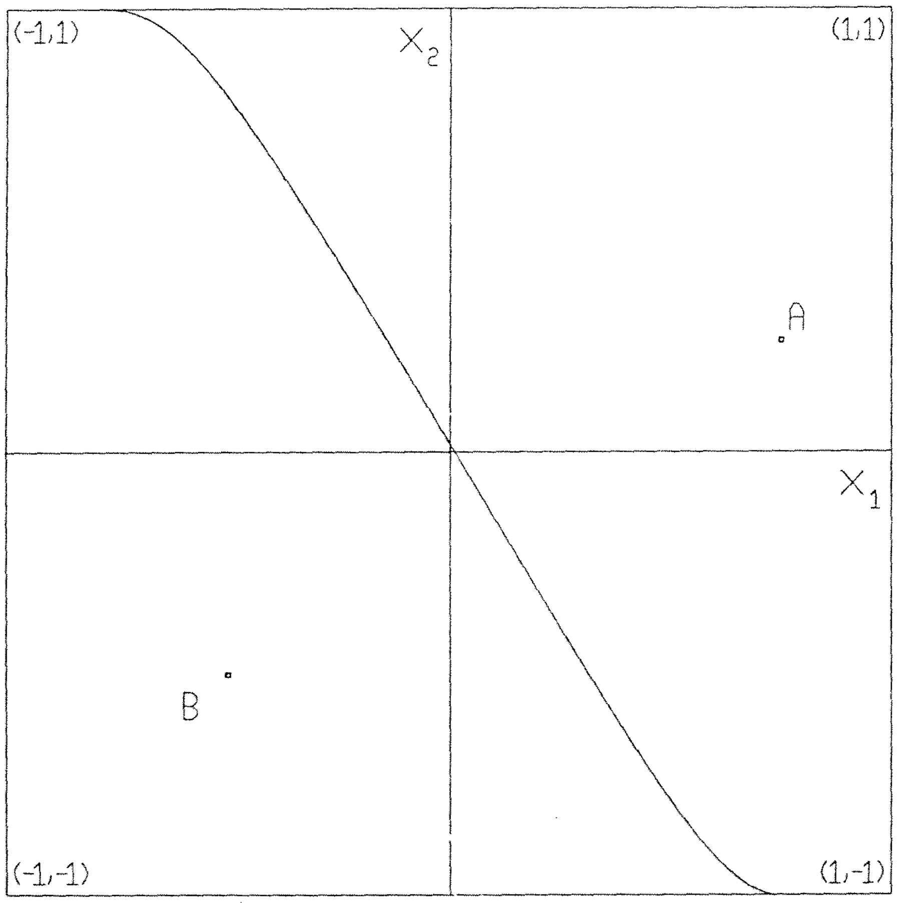

Figure 2.4: A demonstration of the basins of attraction for a network with two hidden neurons and two visible neurons, after training the network to store the vectors $A = (0.25, 0.75)^T$ ans $B = (-0.5, -0.5)^T$.

ponents of the patterns are chosen randomly and set to zero. Fig. (2.6) shows the error fraction versus the number of missing components. The capacity according to Theorem 2.4 for the two-layer network is 11. We have done several experiments with 11 memory vectors. In only a few instances did learning succeed. The reason is more the fact that the augmented saturation layer vectors turn out to be linearly dependent, rather than that the learning algorithm fails to converge to the global minimum. We have also done several experiments with 10 memory vectors. In all our trials learning succeeded. However, the results with 10 memory vectors are quite worse than those in the case of 5 memory vectors. The main reason is that the number of spurious equilibria in the 10 vector case is larger. We have observed in general that when operating close to full capacity, the number of spurious equilibria increases. A reasonable strategy is therefore to operate at half capacity (or less), where the number of spurious equilibria is not high. We performed another experiment using a four-layer network. We have 10 neurons per layer, and 5 memory vectors. The results are shown in Fig. (2.7) for the case of having noisy memory vectors, and in Fig. (2.8) for the case of having memory vectors with missing components. We observe that the results are close to those of the two-layer network. For high signal-to-noise ratios, the two-layer network achieves higher recall accuracy, whereas for low signal-to-noise ratios the four-layer network performs better. We were able to store up to 13 memory vectors using the four-layer network. As in the case of two-layer network, close to the capacity, the performance is not very good.

Finally, the storage capacity of the network for binary vectors is tested. We implemented the method described in Section 2.5. A two-layer network is considered, having 10 neurons in each layer. Of course, in this case there is no saturating hidden layer. Again, 5 memory vectors are considered. The pattern completion ability of the network is tested, and Fig. (2.9) shows the result. We observe that

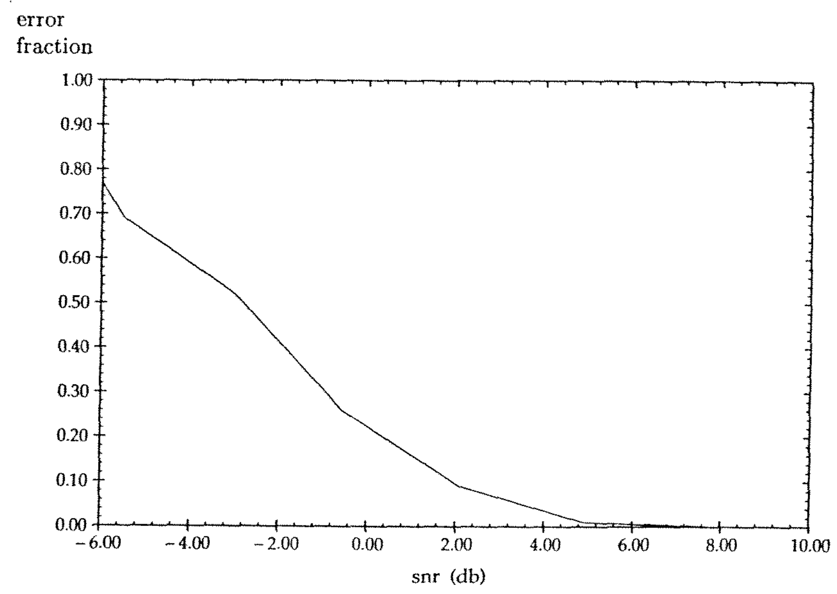

Figure 2.5: The error fraction versus the signal-to-noise ratio for a two-layer network with 10 neurons per layer. The signal-to-noise ratio here means 10 times the logarithm of the ratio of the average of the square of the vector components and the noise variance.

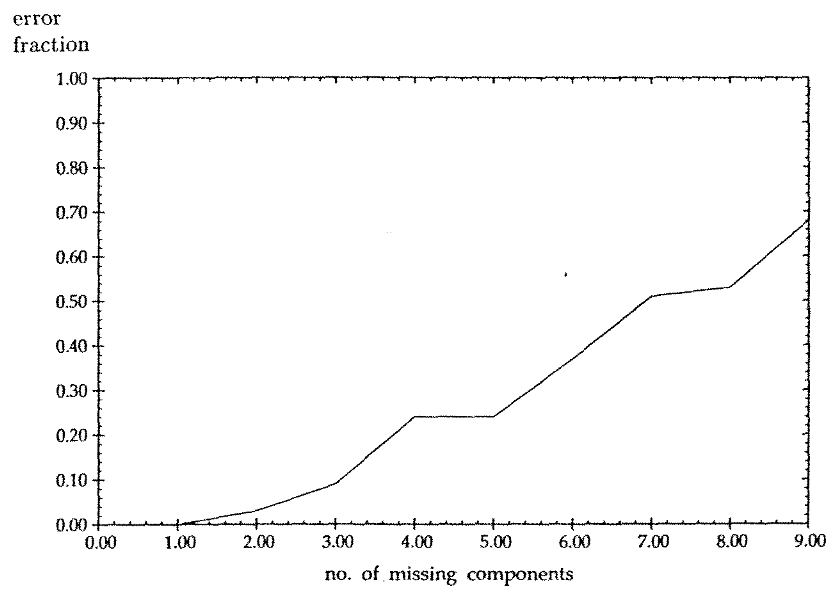

Figure 2.6: The error fraction versus the number of missing components for a two-layer network with 10 neurons per layer.

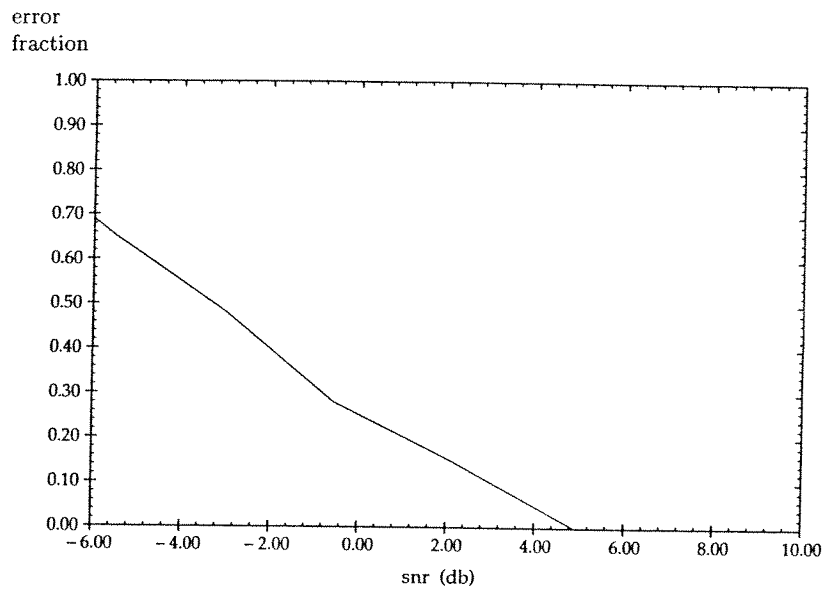

Figure 2.7: The error fraction versus the signal-to-noise ratio for a four-layer network with 10 neurons per layer.

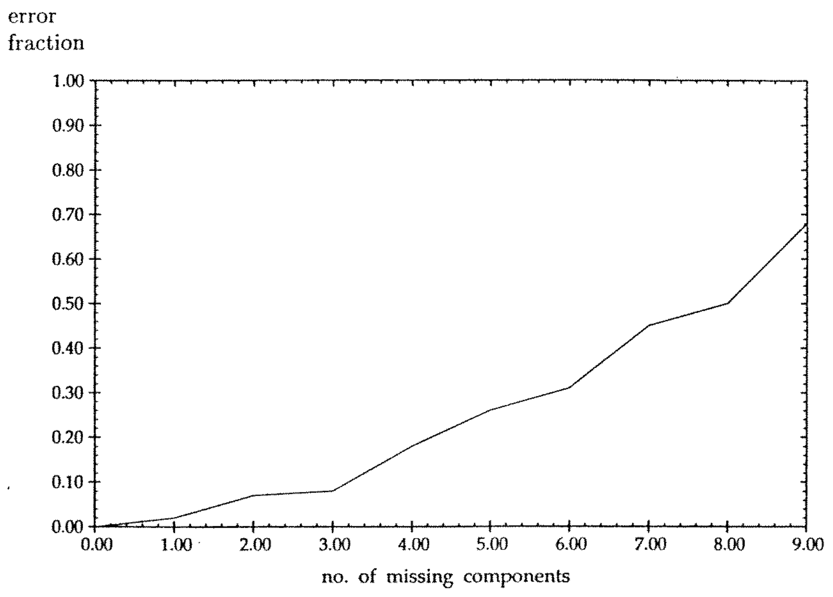

Figure 2.8: The error fraction versus the number of missing components for a four-layer network with 10 neurons per layer.

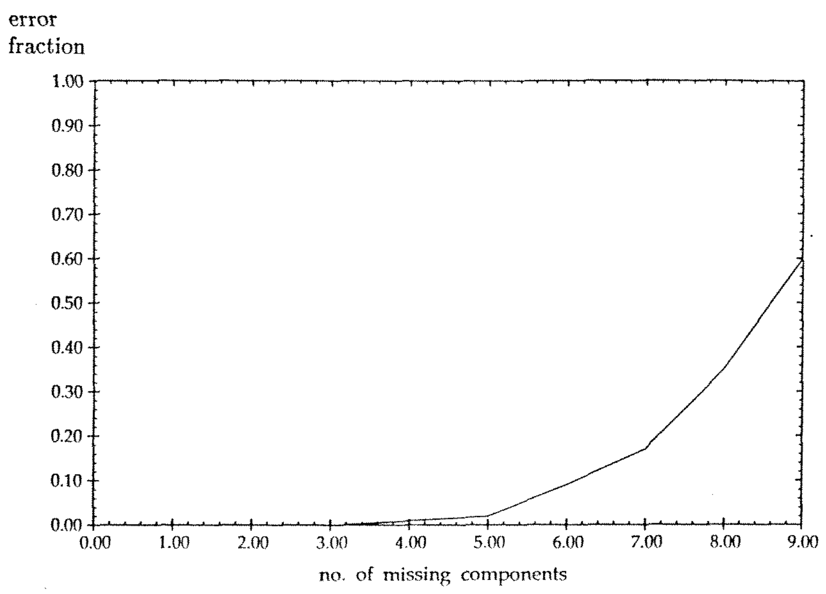

Figure 2.9: The error fraction versus the number of missing bits for a two-layer network with 10 neurons per layer, storing binary vectors.

the performance is better than in the case of storing real valued vectors, because the case of binary vectors is to a greater extent below its capacity, which is here quite higher than in the case of analog vectors.

2.8 CONCLUSION

We have developed a new associative memory model for real valued vectors, using the continuous feedback neural network model. We have considered an architecture consisting of a visible layer and a number of hidden layers connected in a circular fashion. Training is reduced to the problem of training a feedforward network, and therefore the standard backpropagation routines can be used.

The proposed associative memory is of a distributed nature, and therefore a localized fault in one of the neurons will not necessarily "erase" one of the memories, unlike some of the associative memories, for instance, those of the winner-take-all type (see Lippmann's Hamming Net [18], and Majani *et al.* [19]). The strength of the new method is that it can cope with constraints on the architecture, like fan-in and fan-out limitations, or locality of interconnections. In contrast, most of the other associative memory models do not possess such a flexibility, and, for example, they require full connectivity. Another advantage of the new memory is that it can store any given set of vectors. For most of the distributed associative memories (i.e., not counting winner-take-all type), such as those using the Hopfield binary model, there are sets of vectors that cannot be stored (see [20]).

REFERENCES

[1] J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. USA*, vol. 79, pp. 2554-2558, 1982.

[2] S. Venkatesh and D. Psaltis, "Linear and logarithmic capacities in associative neural networks," *IEEE Trans. Inform. Theory*, vol. 35, no. 3, pp. 558-568, 1989.

[3] L. Personnaz, I. Guyon, G. Dreyfus, "Information storage and retrieval in spin-glass like neural networks," *J. Phys. Lett.*, 46: L359-L365.

[4] T. Chiueh and R. Goodman, "A neural network classifier based on coding theory," in *Neural Information Processing Systems*, D. Anderson, *Ed.*, American Institute of Physics, New York, NY, 1988, pp. 174-183.

[5] J.-W. Wong, "Recognition of general patterns using neural networks," *Biological Cybernetics*, vol. 58, no. 6, pp. 361-372, 1988.

[6] A. Michel and J. Farrell, "Associative memories via artificial neural networks," *IEEE Control Systems Mag.*, pp. 6-17, April 1990.

[7] J. Hopfield, "Neurons with graded response have collective computational properties like those of two state neurons," *Proc. Nat. Acad. Sci. USA*, vol. 81, pp. 3088-3092, 1984.

[8] J. Farrell and A. Michel, "A synthesis procedure for Hopfield's continuous-time associative memory," *IEEE Trans. Cicuits Syst.*, Vol. 37, No. 7, pp. 877-884, 1990.

[9] F. Pineda, "Dynamics and architecture in neural computation," *Journal of Complexity*, Sept. 1988.

[10] A. Atiya and Y. Abu-Mostafa, "A method for the associative storage of analog vectors," in *Advances in Neural Information Processing Systems 2*, D. Touretzky, *Ed.*, Morgan Kaufmann Publishers, San Mateo, CA, 1989, pp. 590-595.

[11] S. Grossberg and M. Cohen, "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks," *Trans. Syst., Man, Cybernetics*, Vol. SMC-13, pp. 815-826, 1983.

[12] J. Guckenheimer and R. Holmes, *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*, Springer-Verlag, 1983.

[13] R. Abraham, J. Marsden, and T. Ratiu, *Manifolds, Tensor Analysis and Applications*, Addison Wesley, Reading, MA, 1983.

[14] B. Kosko, "Adaptive bidirectional associative memories," *Applied Optics*, Vol. 26, No. 23, pp. 4947-4960, 1987.

[15] B. Kosko, "Bidirectional associative memories," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-18, no. 1, pp. 49-60, 1988.

[16] J. Franklin, *Matrix Theory*, Prentice-Hall, Englewood Cliffs, New Jersey, 1968.

[17] A. Guez, V. Protopopsecu, and J. Barhen, "On the stability, storage capacity, and design of nonlinear continuous neural networks," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-18, no. 1, pp. 80-87, 1988.

[18] R. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, vol. 4, no. 2, pp. 4-22, 1987.

[19] E. Majani, R. Erlanson, and Y. Abu-Mostafa "On the K-winners-take-all network," in *Advances in Neural Information Processing Systems I*, D. Touretzky, *Ed.*, Morgan Kaufmann Publishers, San Mateo, CA, 1988, pp. 634-642.

[20] J. Bruck and J. Sanz, "study on neural networks," *International Journal of Intelligent Systems*, Vol. 3, pp. 59-75, 1988.

CHAPTER 3

AN UNSUPERVISED LEARNING METHOD FOR NEURAL NETWORKS

3.1 INTRODUCTION

One of the most important features in neural networks is its learning ability, which makes it in general suitable for computational applications whose structure is relatively unknown. Pattern recognition is one example of such kinds of problems. For pattern recognition there are mainly two types of learning: supervised and unsupervised learning (refer to Duda and Hart [1]). Supervised learning means that examples of the patterns as well as their class identities are known. Unsupervised learning, on the other hand, means that only examples of the patterns are available. The information about the class membership of the patterns is not given, either because of lack of knowledge or because of the high cost of providing class labels associated with each of the training patterns. We might wonder how we can hope to design a pattern classifier without the knowledge of the class membership of the training patterns. Fortunately, in typical pattern recognition problems patterns from the same class tend to have similarities, whereas patterns from different classes have relatively large differences. Therefore, a tentative way for designing the classifier is to detect the groups of patterns possessing similarities.

A pattern is represented by a (say, N-dimensional) vector of values representing features extracted from the pattern, the simplest of which in case of a time-signal are its samples at regularly spaced time instants, or in case of an image, are the pixel values. Thus, a pattern represents a point in an N-dimensional space. Consider a set of training patterns. The pattern vectors tend to form clusters of points in the N-dimensional space, a cluster for each class. This is because patterns from the same class tend to be similar and therefore they will be close in distance in the N-dimensional space. Therefore, detecting the clusters is the first step in typical unsupervised learning methods.

The clusters are usually separated by regions of low pattern density. We present here a new method for unsupervised learning using neural networks (see also Atiya [2]). The basic idea of the method is to update the weights of the neurons in a way to move the decision boundaries in places sparse in patterns. This is one of the most natural ways to partition clusters. It is more or less similar to the way humans visually decompose clusters of points in three or less dimensions. This way contrasts with the traditional approaches of unsupervised learning, such as the ISODATA algorithm (Duda and Hart [1]), as well as neural network approaches such as the competitive learning [3] and the self-organizing maps [4], whereby the estimation of the membership of a pattern vector depends only on the the distances between this vector and suitably estimated class representative vectors.

3.2 THE MODEL

Let $\mathbf{x}^{(1)}, ..., \mathbf{x}^{(M)}$ represent the training pattern vectors. Let the dimension of the vectors be N. The training patterns represent several different classes. The class identities of the training patterns are not known. Our purpose is to estimate the number of classes available as well as the parameters of the neural classifier. Let us first consider the case of having two classes. Our neural classifier consists of one neuron (see Fig. 3.1) with N inputs corresponding to the N components of the pattern vector $\mathbf{x}$ to be classified. The output of the neuron is given by

$$y = f(\mathbf{w}^T \mathbf{x} + w_o),$$

where $\mathbf{w}$ is a weight vector, w_o is a threshold and f is a sigmoid function from -1 to 1 with $f(0) = 0$; e.g.,

$$f(u) = \tanh(u).$$

Positive output means class 1, negative output means class 2 and zero output means undecided. Since the output can cover the full range from -1 to 1, we can interpret it as indicating the degree of membership of the pattern to each of the two classes. For example, if the output is very near 1, then the pattern with high probability belongs to class 1. An output near zero means that the pattern membership is uncertain and the two classes are candidates with close likelihood. This type of classifier is called fuzzy classifier.

The classifier partitions the pattern space into two decision regions: the decision region for class 1, given by the half-space $\{\mathbf{x}|\mathbf{w}^T\mathbf{x} + w_o > 0\}$; and the decision region for class 2, given by the other half-space $\{\mathbf{x}|\mathbf{w}^T\mathbf{x} + w_o < 0\}$. The decision boundary is the boundary between the two decision regions. It is given by the hyperplane

$$\mathbf{w}^T\mathbf{x} + w_o = 0$$

(see Fig. 3.2). Patterns near the decision boundary will produce outputs close to zero, while patterns far away from the boundary will give outputs close to 1 or -1. Assuming $\|\mathbf{w}\| \leq a$, where a is a constant, a good classifier is one that produces an output close to 1 or -1 for the training patterns, since this indicates its "decisiveness" with respect to the class memberships. Therefore, we design the classifier so as to maximize the criterion function:

$$J = \frac{1}{M}\sum_j f^2(\mathbf{w}^T\mathbf{x}^{(j)} + w_o) - \left[\frac{1}{M}\sum_j f(\mathbf{w}^T\mathbf{x}^{(j)} + w_o)\right]^{2q},$$

subject to $\|\mathbf{w}\| \leq a$, where q is a positive integer (for best results we take q=2). The first term is a measure of the decisiveness of the classifier with the given test pattern set. Regarding the second term, it has the following purpose. The first

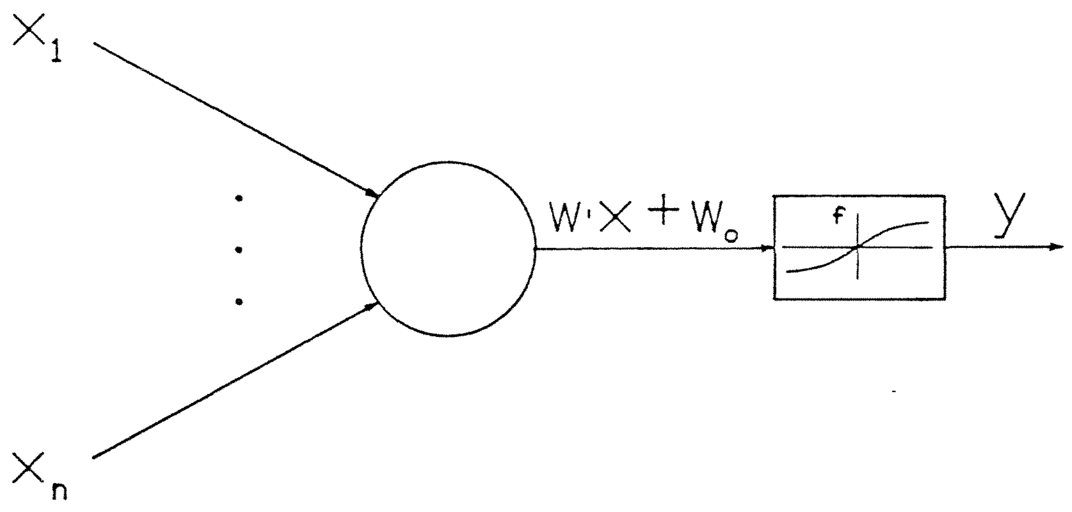

Fig. 3.1:The model of a first order neuron with a sigmoid-shaped function.

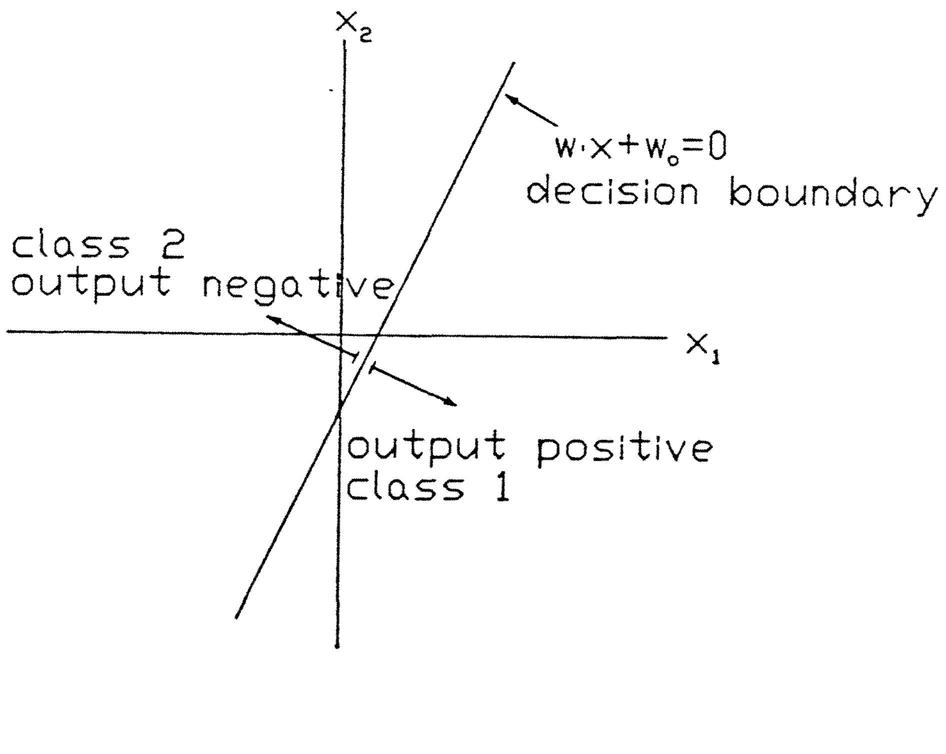

Fig. 3.2: The decision regions and the decision boundary of a first order neuron.

term is maximized if the decision hyperplane is very far away from the patterns, resulting in the output being very close to 1 for all test patterns (or close to -1 for all patterns); i.e., all patterns are assigned to one class only, which is a trivial solution. Incorporating the second term will prevent this trivial solution because if $f(\mathbf{w}^T\mathbf{x}^{(j)} + w_o) \simeq 1$ for all j (or -1 for all j), then J will be nearly zero. However, J is non-negative for any $\mathbf{w}$ and w_o because of the following:

$$\begin{aligned} J &= \frac{1}{M}\sum_j f^2(\mathbf{w}^T\mathbf{x}^{(j)} + w_o) - (\bar{f})^{2q} \\ &\geq \frac{1}{M}\sum_j f^2(\mathbf{w}^T\mathbf{x}^{(j)} + w_o) - (\bar{f})^2, \end{aligned}$$

where

$$\bar{f} = \frac{1}{M}\sum_j f(\mathbf{w}^T\mathbf{x}^{(j)} + w_o).$$

The inequality follows because $\bar{f}$, the average of the outputs, is less than one in magnitude and $q \geq 1$. Then we get

$$J \geq \frac{1}{M}\sum_j \left[f(\mathbf{w}^T\mathbf{x}^{(j)} + w_o) - \bar{f}\right]^2 \geq 0$$

and hence the trivial solution of assigning all patterns to one class only is ruled out because it corresponds to a minimum, not a maximum, for J.

We iterate the weights in a steepest ascent manner in an attempt to maximize the defined measure,

$$\begin{aligned} \Delta\mathbf{w} &= \rho\frac{\partial J}{\partial \mathbf{w}} \\ &= \rho\frac{2}{M}\sum_j \left[f(\mathbf{w}^T\mathbf{x}^{(j)} + w_o) - q(\bar{f})^{2q-1}\right] f'(\mathbf{w}^T\mathbf{x}^{(j)} + w_o)\mathbf{x}^{(j)} \end{aligned}$$

and

$$\begin{aligned} \Delta w_o &= \rho\frac{\partial J}{\partial w_o} \\ &= \rho\frac{2}{M}\sum_j \left[f(\mathbf{w}^T\mathbf{x}^{(j)} + w_o) - q(\bar{f})^{2q-1}\right] f'(\mathbf{w}^T\mathbf{x}^{(j)} + w_o), \end{aligned}$$

where ρ is the step size. The iteration is subject to the condition $\|\mathbf{w}\| \leq a$. Whenever, after some iteration, this condition is violated, $\mathbf{w}$ is projected back to the surface of the hypersphere $\|\mathbf{w}\| = a$ (simply by multiplying by $a/\|\mathbf{w}\|$). The term $f'(\mathbf{w}^T\mathbf{x}^{(j)}+w_o)$ in the update expression gives the patterns near the decision boundary more effect than the patterns far away from the decision boundary. This is because $f'(u)$ is high whenever u is near zero and goes to zero when the magnitude of u is large. Thus, what the method is essentially doing is iterating the weights in such a way as to move the decision boundary to a place sparse in patterns. The relative importance of patterns near the decision boundary increases for large a. This is because $\mathbf{w}$ tends to go to the surface of the sphere $\|\mathbf{w}\| \leq a$. Large a will therefore result in the argument of f' being in general higher than for the case of small a, thus resulting in a smaller strip with high f' around the decision boundary. Of course, without the constraint $\mathbf{w}$ could grow in an unbounded fashion, resulting in $f(\mathbf{w}^T\mathbf{x}^{(j)} + w_o)$ being very near 1 or -1 for all j for most possible partitions, and we obtain therefore possibly bad solutions.

3.3 EXTENSIONS

We have shown in the previous section a method for training a linear classifier. To extend the analysis to the case of a curved decision boundary, we use a higher-order neuron; i.e., the output is described by

$$y = f(w_o + \sum_i w_i x_i + \cdots + \sum_{i_1 \leq \cdots \leq i_L} w_{i_1 \cdots i_L} x_{i_1} \cdots x_{i_L}),$$

where x_i denotes the i^{th} component of the vector $\mathbf{x}$ and L represents the order. Higher-order networks have been investigated by several researchers, refer for example to Psaltis and Park [5], Psaltis *et al.* [6] and Chen *et al.* [7]. They achieve simplicity of design while still having the capability of producing fairly sophisticated non-linear decision boundaries.

The decision boundary for the higher-order neuron is given by the equation

$$w_o + \sum_i w_i x_i + \cdots + \sum_{i_1 \leq \cdots \leq i_L} w_{i_1 \cdots i_L} x_{i_1} \cdots x_{i_L} = 0.$$

The higher-order case is essentially a linear classifier applied to augmented vectors of the form

$$(x_1, \cdots, x_N, \cdots, x_1^L, x_1^{L-1} x_2, \cdots, x_N^L)$$

(the superscripts denote here powers), using an augmented weight vector

$$(w_1, \cdots, w_N, \cdots, w_{1\cdots 11}, w_{1\cdots 12}, \cdots, w_{N \cdots NN}).$$

We can therefore apply basically the same training procedure described for the linear classifier case on the set of augmented vectors; i.e., we update the weights according to

$$\Delta w_{i_1 \cdots i_l} = \rho \frac{2}{M} \sum_j \left[f(u_j) - q(\bar{f})^{2q-1} \right] f'(u_j) x_{i_1}^{(j)} \cdots x_{i_l}^{(j)}$$

$$\Delta w_o = \rho \frac{2}{M} \sum_j \left[f(u_j) - q(\bar{f})^{2q-1} \right] f'(u_j),$$

where $f(u_j)$ is the output of the neuron when applying the pattern $\mathbf{x}^{(j)}$; i.e.,

$$u_j = w_o + \sum_i w_i x_i^{(j)} + \cdots + \sum_{i_1 \leq \cdots \leq i_L} w_{i_1 \cdots i_L} x_{i_1}^{(j)} \cdots x_{i_L}^{(j)}$$

and

$$\bar{f} = \frac{1}{M} \sum_j f(u_j).$$

We have considered so far the two-class case. The extension to the multi-class case is as follows. We apply the previously described procedure (preferably the curved boundary one), i.e., partition the patterns into two groups. One group of patterns S^+ corresponds to positive outputs and represents an estimate of a collection of classes. The other group S^- corresponds to negative outputs and represents an estimate of the remaining classes. Then we consider the group S^+ separately (i.e., we use only the patterns in S^+ for training), and partition it further into two groups S^{++} and S^{+-} corresponding to positive and negative outputs, respectively. Similarly, we partition S^- into S^{-+} and S^{--}. We continue in this hierarchical manner until we end up with the final classifier; see Fig. 3.3 for an example. We stop the partitioning of a group whenever the criterion function J associated with the partition falls below a certain threshold. (One practically observes that J in general decreases slightly after each partition until one cluster remains, whose partitioning results in a relatively abrupt decrease of J.) The final classifier now consists of several neurons, the sign pattern of whose outputs is an encoding of the class estimate of the presented pattern. Note that the output of the neuron responsible for breaking up some group plays a role in the encoding of only the classes contained in that group. Therefore, the encoding of a class could have a number of "don't cares". Refer to the next section for a constructive example of the extension to the multi-class case.

3.4 IMPLEMENTATION EXAMPLES

The new method is implemented on two two-class problems and a five-class problem. In all examples the patterns of each class are generated from bivariate Gaussian distributions. The first example is a two-class problem with a large overlap between the two classes. Fig. 3.4 shows the resulting decision boundary when applying the new method using a first-order neuron. One observes that the obtained partition agrees to a large extent to that a human would estimate when attempting to decompose the two clusters visually. Regarding the second example, we have two classes with equal diagonal covariance matrices whose diagonal entries differ much, resulting in two long clusters. Fig. 3.5 shows the results of the application of the proposed method using a first-order neuron. In another example we have a

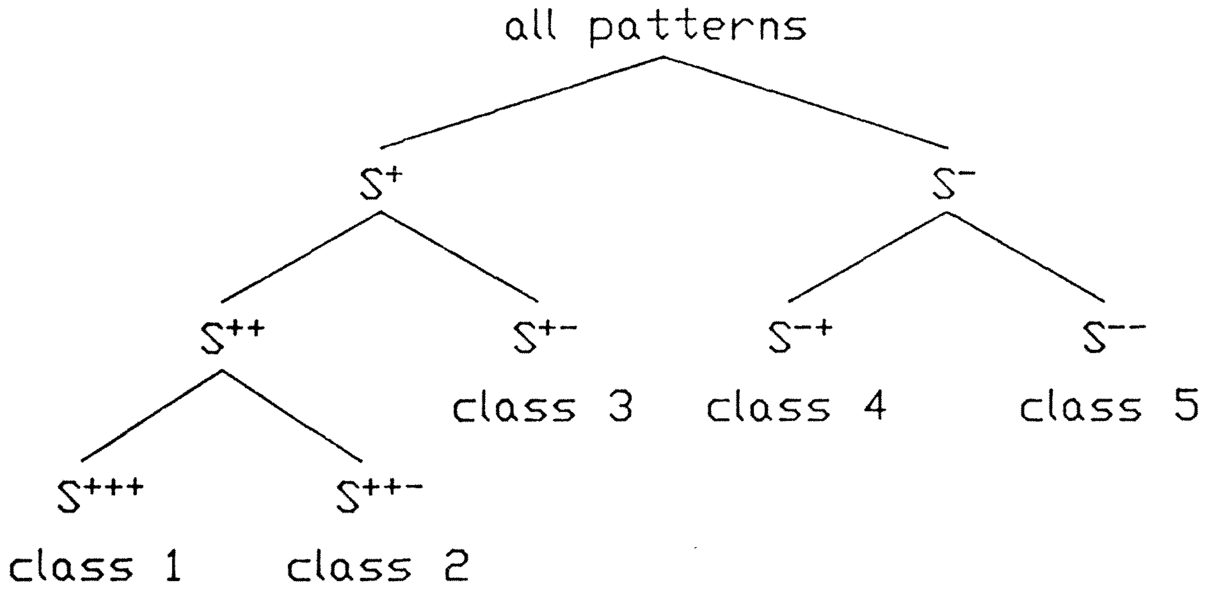

Fig. 3.3: An example of the hierarchical design of the classifier.

five-class problem with the means being on the corners and the center of a square. We used third-order neurons. Fig. 3.6 shows the results. We ended up with four neurons partitioning the patterns. The first neuron is responsible for partitioning the whole collection of patterns into the two groups S^+ and S^-, separated by the boundary B_1; the second neuron partitions the group S^+ into the groups S^{++} and S^{+-}, separated by the boundary B_2; the third neuron divides S^{++} into S^{+++} and S^{++-} with boundary B_3; finally, the fourth neuron partitions S^- into S^{-+} and S^{--}, the boundary being B_4 (see also Fig. 3.3). One observes that the method partitioned the patterns successfully. As a pattern classifier, the four neurons give the encoding of the class estimate of the presented pattern. The codes of the five classes are $++{+X}$, $++-X$, $+-XX$, $-XX+$, and $-XX-$, where the $+$'s and $-$'s represent the signs of the outputs and the X means "don't care". In the last simulation experiment we test the performance of the method on a 15-dimensional example having 30 clusters. The means of the clusters are generated using a uniform distribution in $[-1,1]^{15}$. The inputs are Gaussian and uncorrelated; the standard deviation is 0.15 for each input and for each class. Using second-order neurons, the new method resulted in the correct partitioning of the clusters, the number of neurons used for the partitioning being 28.

We remark that in all the previous problems, the weights are initialized by generating them randomly. A final important comment about the method is that when using a high-order neuron and a constant step size, it is imperative to scale the inputs so that their maximum magnitude is neither much higher nor much lower than 1. The reason is that otherwise, the higher-order terms will respectively either dominate or be dominated by the lower-order terms, and unsatisfactory results might be obtained.

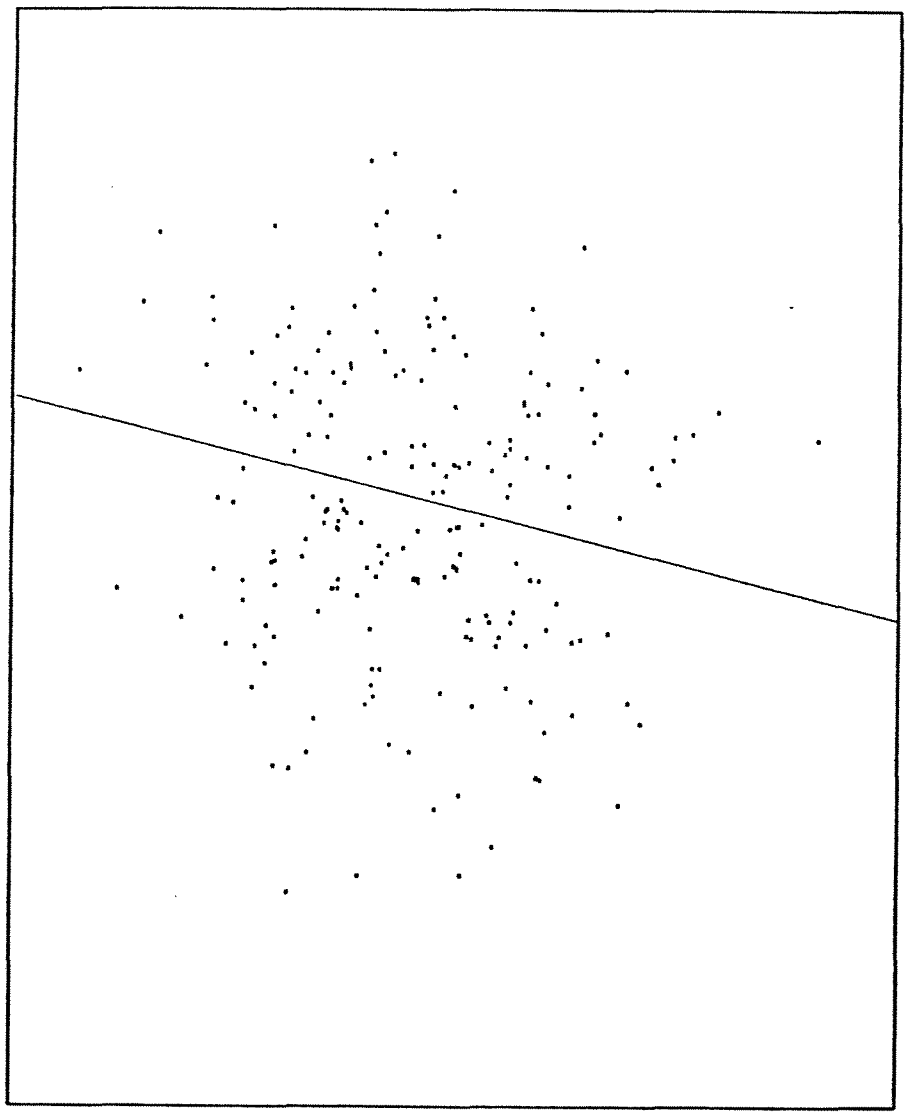

Fig. 3.4: A two-cluster example with the decision boundary of the designed classifier.

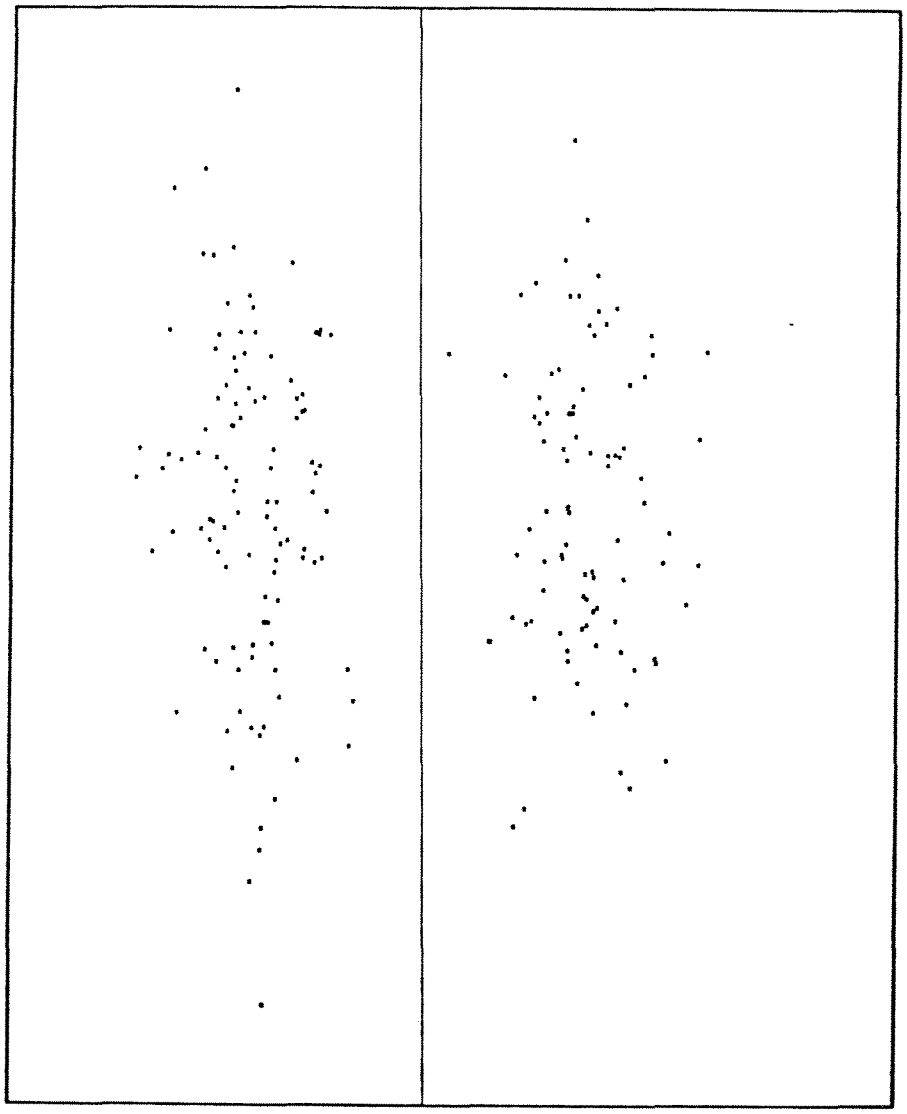

Fig. 3.5: A two-cluster example with the decision boundary of the designed classifier.

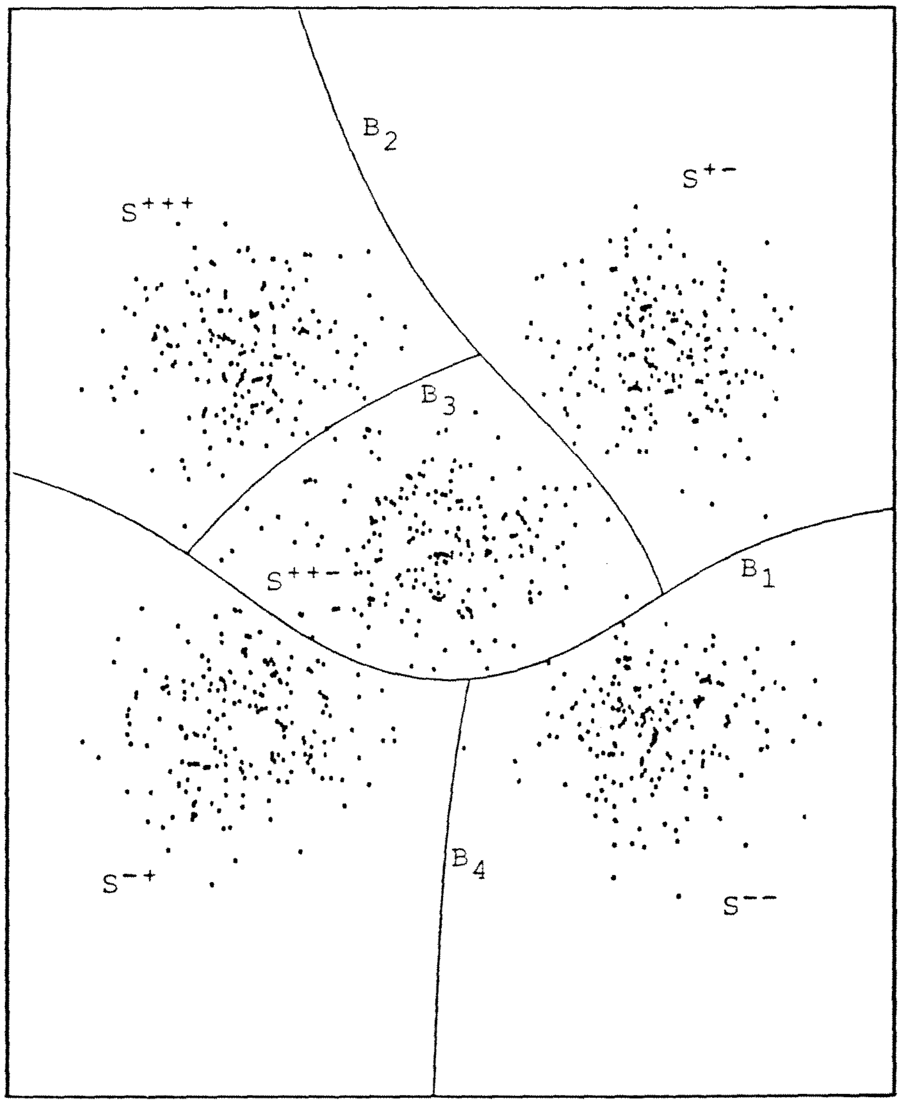

Fig. 3.6: A five-class example showing the decision boundaries of the hierarchically designed classifier. There are four neurons associated with the boundaries B_1, B_2, B_3, and B_4.

3.5 CONCLUSION

A new neural, unsupervised learning technique has been proposed in this work. This technique is based on the hierarchical partition of the patterns. Each partition corresponds to one neuron, which is in general a higher-order neuron. The partition is performed by iterating the neuron weights in an attempt to maximize a defined criterion function. The method is implemented on several examples and is found to give good results. The method is fast, as it takes typically from about 2 to 5 iterations to converge (by one iteration we mean one sweep through the set of training patterns). Although the proposed method is prone to get stuck in local minima, this did happen in our simulations in only very difficult problems, and this problem could be solved by using gradient algorithms for searching for the global maximum, like the Tunneling Algorithm (Levy and Montalvo [8]).

REFERENCES

[1] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, John Wiley and Sons, New York, 1973.

[2] A. Atiya, "An unsupervised learning technique for artificial neural networks," *Neural Networks*, Vol. 3, No. 6, pp. 707-711, 1991.

[3] S. Grossberg, "Adaptive pattern classification and universal recording: Part I. Parallel development and coding of neural feature detectors," *Biological Cybernetics, 23*, pp. 121-134, 1976.

[4] T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag, Berlin, 1984.

[5] D. Psaltis and C. Park, "Nonlinear discriminant functions and associative

memories," J. Denker, Ed., AIP Conf. Proc., Snowbird, Utah, 1986.

[6] D. Psaltis, C. Park and J. Hong, "Higher order associative memories and their optical implementations," *Neural Networks*, Vol. 1, No. 2, pp. 149-163, 1988.

[7] H. Chen, Y. Lee, T. Maxwell, G. Sun, H. Lee, and C. Giles, "High order correlation model for associative memory," J. Denker, Ed., AIP Conf. Proc., Snowbird, Utah, 1986.

[8] A. Levy and A. Montalvo, "The Tunneling Algorithm for the global minimization of functions," *SIAM J. Sci. Stat. Comput.*, Vol. 6, pp. 15-29, January 1985.

CHAPTER 4

OSCILLATIONS IN NEURAL NETWORKS

4.1 INTRODUCTION

The brain is a complex dynamical system. To study its intriguing properties and functions, one has to shed light on the possible role of the different types of behavior of the system. A first step is to consider stable neural networks, i.e., networks where every trajectory tends to an equilibrium. Such networks have been extensively studied and are now relatively well understood (see Hirsch [1]). The next step is to study networks exhibiting oscillatory behavior.

A large variety of oscillations in various parts of the brain have been discovered, see for instance Freeman and Skarda [2], Freeman and Van Dijk [3], Pöppel and Logothetis [4], and the references in Pavlidis [5], to name a few. The idea of analyzing oscillations in neural networks has gained a lot of importance after recent discoveries by Gray *et al.* [6] and Gray and Singer [7] (see also Eckhorn *et al.* [8]), which indicate that oscillations seem to play a role in the extraction of high level features from the visual scene. Before explaining these discoveries in detail, we will give a short introduction about the vertebrate visual system (see also Kuffler *et al* [9]).

The visual system is one of the most studied sensory systems in the vertebrate brain. It consists of several layers. The first few layers are responsible for extracting several simple features in the visual scene, for example, the detection of edges and the detection of motion. Subsequent layers extract progressively higher level features. The outputs of these processing stages are pieced together at higher centers of the brain, to produce such high level tasks as the perception of objects, the understanding of the visual scene, and the understanding of motion in the scene. The first few processing stages are relatively well understood (at least their overall functionality, but maybe not so much the details of how the neural responses are generated by the interactions between the cells). On the other hand, sufficient understanding of the deeper processing stages is still lacking. The first stage in the visual system is the retina. Upon the presentation of a visual stimulus or input, the photoreceptor cells absorb light and produce a proportional response. The subsequent layers in the retina process the visual signal further, and the output is relayed via the optic nerve to the so-called lateral geniculate nucleus (LGN). The optic nerve consists of a bundle of about a million axons of the cells of the last layer of the retina, the ganglion cell layer. The initial processing performed at the retina has been tested by stimulating the retina with a variety of patterns of light of different shapes, sizes, and colors. It has been found that the retinal ganglion cell responds best to a roughly a circular spot of light of a particular size and a particular part of the visual field. (The particular part of the visual field stimulating a particul cell is called the receptive field of the cell.) The LGN is another processing station, whose output goes to the primary visual cortex. From there, the output goes to further destinations such as neighboring cortical areas. There are also feedback connections from the primary visual cortex to the LGN. The LGN also responds mainly to circular spots of light. On the other hand, the primary visual cortex is activated by more complicated patterns. There are two types of cells: simple and complex cells. A simple cell is usually tuned to respond best to a light bar with a particular width and length, and with a particular orientation. The cells are usually arranged in columns; each column corresponds to a particular position in the visual scene. Each column consists of cells, each being tuned to detect a light bar with a particular orientation. The complex cells are specialized to detect moving light bars. Each complex cell produces strongest activity if a light bar with a particular orientation, and moving in a particular direction is presented in the appropriate position of the receptive field.

The discoveries of Gray *et al.* have been made using multi-unit recordings from the cat primary visual cortex. (Multi-unit recording is a way to record the activity of groups of neurons; a detailed discussion of this method is given next chapter.) Upon the presentation of a moving light bar of suitable orientation and direction, populations of adjacent neurons within a cortical column exhibit an oscillatory response with a frequency of 40-50 Hz. The interesting fact is that zero phase locked oscillations could occur in two remote columns of non-overlapping receptive fields with similar orientation preferences, as much as 7 mm apart. When two short light bars moving in opposite directions are presented as stimuli in the receptive fields of the two cortical sites, then oscillations occurred, but no synchronization is observed. In contrast, when an elongated light bar covering the two receptive fields is presented, the two sites engage in synchronous oscillations.

Thus it seems that synchronous oscillations are a means of linking related features in different parts of the visual scene. It serves as a way for coding global features of the scene. We therefore believe that feedback signals from higher centers of the brain are also involved. The stimulus-specific phase lockings discovered could be the visible part of a more complex coding mechanism for high level features in the visual scene involving all parameters of the oscillations such as phase, frequency, and amplitude. A possible mechanism is the labeling hypothesis, closely related to several previous investigations, for example, by von der Malsburg [10],[11], and von der Malburg and Schneider [12], Abeles [13], and Crick [14]. In the labeling hypothesis neighboring groups of neurons are labeled by certain labels, such as the phase of the oscillation (frequency and/or amplitude). These labels reflect different properties of the image, and the computations involved are the interactions between the labels of the different groups. For example, an image could be segmented by attaching a specific phase to the oscillations of aggregates of neurons associated with each object. (Image segmentation means partitioning the image into regions, each associated with a meaningful object in the scene.) Another example of the labeling hypothesis is to have a frequency of oscillation expressing the speed of motion for the groups of neurons associated with each moving object in the scene. (In fact, Eckhorn *et al.* [8] observed that the frequency of oscillation increases with the increase of the speed of the moving light bar.) Baldi *et al.* [15] show how a two-dimensional field of coupled neural oscillators can be partitioned into patches of roughly the same phase. Some other processing involving other layers with feedback might adjust each patch to the site of one of the objects in the scene, thus leading to the segmentation of the image. In this chapter we are going to study oscillations in various neural network architectures (see also Baldi and Atiya [16]). We will investigate conditions for phase locking of an aggregate of neurons, and we will study how a stimulus applied to two populations of neurons can lead to 0-phase locking.

4.2 THE MODELS

To study oscillations and phase locking phenomema, it is essential to use a model that matches reality as closely as possible. However, because of lack of sufficient biological information (for instance, detailed information about the connectivity), it makes not much sense at this stage to use such detailed models as the one based on Hodgkin-Huxley Equations [17] (the Hodgkin-Huxley Equations are a detailed model for the neuron, explaining the spike generation and propagation process). Also, too much detail and complexity in a model might obscure the underlying principles and mechanisms of oscillations. We are interested in simple and general robust principles, which can serve as guidelines also in studying other oscillation and phase locking phenomena detected in the brain. We have therefore decided to use the Hopfield continuous model [18]:

$$\frac{du_i}{dt} = -\frac{u_i}{\tau_i} + \sum_{j=1}^{N} w_{ij} f_{\lambda_j}(u_j), \qquad x_i = f_{\lambda_i}(u_i) \qquad i = 1, ..., N, \tag{4.1}$$

where, as usual, u_i is the potential of neuron i, x_i is the short time average firing rate, $\mathbf{W} = [w_{ij}]$ is the matrix of synaptic weights and τ_i is the time constant of neuron i. The function f_{λ_i} is a sigmoid function, and λ_i is the gain, which is a parameter controlling the steepness of the sigmoid function. In this chapter we take $f_{\lambda_i}(u_i) = \tanh(\lambda_i u_i)$.

The results of Gray *et al.* seem to indicate that some form of partial organization occurs even at a lower level, namely, that of single neuronal spikes. Therefore, we have also examined oscillations in networks of spiking neurons (similar, for instance, to those considered by Partridge [19] and Knight [20]). In such a model a neuron integrates the incoming inputs until the neuron potential reaches a fixed threshold. At that instant the neuron produces a spike, provided that it is not in the refractory period. The details and the results of this model are presented in Section 4.9.

Consider again the Hopfield continuous model (4.1). It has been studied by many researchers (see Hirsch [1] for an overview). Its stability properties have been examined in detail. For instance, a symmetric weight matrix $\mathbf{W}$ (Grossberg and Cohen [21], and Hopfield [18]) guarantees stability. Another interesting result by Hirsch [22], is that a network satisfying the even loop property is almost stable (i.e., the set of initial states whose trajectories do not converge is of measure zero). The even loop property means that if the network is modeled as a directed graph, then any cycle contains an even number of inhibitory connections.

We present here another new stability result. If the weight matrix is anti-symmetric (i.e., $w_{ij} = -w_{ji}$, $w_{ii} = 0$), then the network is stable. The proof is as follows. Consider the Lyapunov function:

$$G(u_1, ..., u_N) = \sum_{i=1}^{N} \int_0^{u_i} f_{\lambda_i}(u_i) du_i.$$

Clearly $G \geq 0$, $G \to \infty$ as $||\mathbf{u}|| \to \infty$, and

$$\frac{dG}{dt} = \sum_{i=1}^{N} f_{\lambda_i}(u_i) \frac{du_i}{dt} = \sum_{i=1}^{N} -\frac{u_i}{\tau_i} f_{\lambda_i}(u_i) + \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} f_{\lambda_i}(u_i) f_{\lambda_j}(u_j).$$

Since $\mathbf{W}$ is antisymmetric, the second term in the right hand side of the previous equation vanishes. Since $u_i f_{\lambda_i}(u_i) > 0$, the time derivative of G is less than or equal to 0, with equality if and only if $u_i = 0$, for all i. Therefore, the system converges to the origin, which is the unique equilibrium for the network.

Little work has been done, though, to analyze the dynamical properties of the model. Baird [23] and [24] studied Hopf bifurcations in oscillatory networks, for the purpose of analyzing the olfactory bulb. Li and Hopfield [25] studied oscillations in a two-layer network, also in their modeling of the olfactory bulb.

4.3 ON NON-LINEAR OSCILLATIONS

Because of the ubiquity of oscillations in such diverse fields as biology, chemistry, ecology, economics, and engineering, the theory of non-linear oscillations has attracted the attention of many researchers since as far back as last century. Oscillations, particularly in two-dimensional systems, are to some extent well understood, and a number of theoretical results exists. (Oscillations cannot occur in one-dimensional flows, unless the system is driven by an external periodic driving force.) One of the well-known theorems on the existence of oscillations in two-dimensional systems is the Poincaré-Bendixson theorem. It states that if a trajectory remains in some compact region D, and if it does not approach any equilibrium point in D, then it converges to a periodic orbit, which lies entirely in D. Another interesting result, reported by Bendixson, states that if the divergence of the vector field is positive everywhere (or negative everywhere) on a simply connected region, then no periodic orbit can lie entirely in the region.

For high-dimensional systems, on the other hand, there are only very few theorems on the existence of periodic orbits, the most important of which is the Hopf bifurcation theorem. Let λ be a parameter vector of the system. If at $\lambda = \lambda^*$ several conditions are satisfied, the main condition being that the Jacobian corresponding to an equilibrium possesses a pair of pure imaginary eigenvalues (see Hassard *et al.* [26]), then the point $\lambda = \lambda^*$ is said to be a Hopf bifurcation point. The equilibrium bifurcates into a periodic orbit. This theorem, however, only proves the existence of a periodic orbit immediately beyond the bifurcation point $\lambda = \lambda^*$, but does not specify how far beyond it the periodic orbit will still exist. Nevertheless, the theorem has been of practical importance.

Several results exist concerning the approximation of the frequency and the amplitude of the oscillations (see Nayfeh and Mook [27]). As a first approximation, we can assume that the oscillations are small and close to sinusoidal, and linearize the system. A better approximation can be obtained by considering additional higher-order terms in the Taylor series expansion of the nonlinearities of the vector field. This allows us to obtain several harmonics of the oscillations. This latter approach is usually difficult for high-dimensional systems, as we obtain a simultaneous set of N non-linear algebraic equations, which are usually hard to solve (N is the dimension). For simplicity, therefore, we have adopted the first approach in our work.

As a first step in understanding the oscillations in neural systems, we will study the oscillations in two-neuron networks.

4.4 TWO-NEURON NETWORK

A two-neuron network with no self-connections (0 diagonal weight matrix) cannot have a stable periodic orbit. (A stable periodic orbit is defined as a periodic orbit that attracts all solutions starting in some open neighborhood containing the periodic orbit.) To prove this fact one can use Hirsch's result for the case $w_{12}w_{21} > 0$. For the case $w_{12}w_{21} < 0$, we use the Lyapunov function

$$G(u_1, u_2) = |w_{12}| \int_0^{u_2} f_{\lambda_2}(u_2) du_2 + |w_{21}| \int_0^{u_1} f_{\lambda_1}(u_1) du_1.$$

A trivial calculation shows that $\frac{dG}{dt} \leq 0$ with equality if and only if $u_1 = u_2 = 0$. Therefore, all trajectories converge to the origin and there are no periodic orbits. Therefore, to achieve oscillations we have to add self-connections. Consider then the system associated with the matrix $w_{11} = w_{22} = w_{12} = -w_{21} = 1$ and assume for simplicity that $\lambda_1 = \lambda_2 = \lambda$ and $\tau_1 = \tau_2 = \tau$.

Proposition 4.1: For the system

$$\begin{aligned} \frac{du_1}{dt} &= -\frac{u_1}{\tau} + f_\lambda(u_1) + f_\lambda(u_2) \\ \frac{du_2}{dt} &= -\frac{u_2}{\tau} - f_\lambda(u_1) + f_\lambda(u_2) \end{aligned} , \tag{4.2}$$

the origin is the only equilibrium point. If $\lambda\tau \leq 1$, the origin is stable and all the trajectories converge to the origin (no cycles). If $\lambda\tau > 1$, the origin is unstable and there exists always a periodic orbit surrounding the origin.

Proof: Equilibrium points of (4.2) satisfy the equations

$$\begin{aligned} h(u_1) &= u_1 - u_2 \\ h(u_2) &= u_1 + u_2 \end{aligned} , \tag{4.3}$$

where $h(u) = 2\tau f_\lambda(u)$. If one of the variables satisfying (4.3) is non-zero, so must the other one be. Moreover, h is odd, i.e. $h(-u) = -h(u)$. Therefore, we need to consider only solutions in the two quadrants defined by $u_1 \geq 0$. We have two cases:

Case 1: $2\lambda\tau \leq 1$ (i.e. $h(u) < u$ for $u > 0$). If $u_2 > 0$, the second equation of (4.3) is violated and if $u_2 < 0$, the first one cannot be satisfied.

Case 2: $2\lambda\tau > 1$ (i.e., there exists a unique point $c > 0$ such that $h(c) = c$ and $h(u) > u$ for $0 < u < c$ and $h(u) < u$ for $u > c$). Clearly, we cannot have $u_1 = c$ (respectively, $u_2 = c$) for then $u_2 = 0$ (respectively, $u_1 = 0$). Four subcases need to be examined depending on the position of u_2 with respect to c and $-c$. If $u_2 > c$, then obviously the second equation of (4.3) cannot be satisfied. If $0 < u_2 < c$, then from the second equation we have $u_1 = h(u_2) - u_2 < c$. Therefore, $h(u_1) > u_1$ and this contradicts the first equation. The remaining cases are similar, and as a result the origin is the unique equilibrium point.

We can calculate the eigenvalues β of the Jacobian at the origin to obtain the expression $\beta = (\tau\lambda - 1)\tau^{-1} \pm i\lambda$. As a consequence, if $\lambda\tau > 1$, the origin is unstable, and if $\lambda\tau < 1$, the origin is stable. Now, if $\lambda\tau \leq 1$, we can prove, using Bendixson's result (see previous section), that the system cannot have any periodic trajectories, for the divergence of the vector field defined by (4.2) is given by $-2/\tau + f'_\lambda(u_1) + f'_\lambda(u_2)$, which is always strictly negative if $\lambda\tau \leq 1$ (except at the origin when $\lambda\tau = 1$, in which case the divergence is 0). If $\lambda\tau > 1$, we have seen that the origin is unstable, with both eigenvalues having positive real parts. Therefore, it is possible to draw a small enough closed curve around the equilibrium such that the vector field at this curve points outwards. Also, we can easily show that there exists a very large closed curve such that the vector field points inwards. Hence by Poincaré-Bendixson's theorem there exists a periodic trajectory in the region in between the small and the large curve. ∎

4.5 RING ARCHITECTURE

We begin by studying the ring architecture (see Figure 4.1). The neurons are arranged in a ring, with connections running in one direction. It will be seen in the next section that many properties of this simple network carry over to the case of layered networks with feedback, an architecture possibly often encountered in the nervous system. The equations for the ring are

$$\frac{du_i}{dt} = -\frac{u_i}{\tau_i} + w_{i,i-1} f_{\lambda_{i-1}}(u_{i-1}) \equiv F_i(u_i, u_{i-1}), \qquad x_i = f_{\lambda_i}(u_i) \qquad i = 1, ..., N, \tag{4.4}$$

where $w_{10} \equiv w_{1N}$. We restrict ourselves to rings at least 3 in length. We have seen in the last section that the two-neuron ring cannot oscillate. Let $A = \Pi_{i=1}^{N} w_{i,i-1}$, and $\Lambda = \Pi_{i=1}^{N} \lambda_i$.

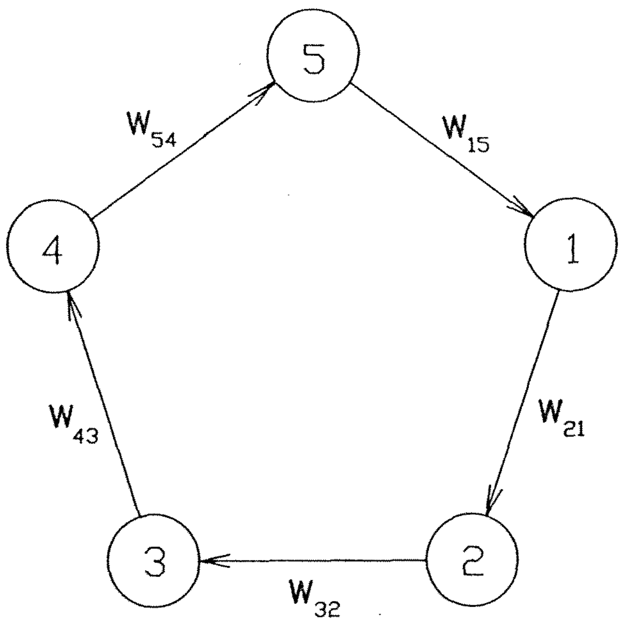

Figure 4.1: A 5-neuron ring

By Hirsch's theorem [22], an even number of inhibitory connections in the ring ($A > 0$) leads to convergent behavior. Let us now consider the case of an odd number of inhibitory connections ($A < 0$). For simplicity, let the time constants be equal ($\tau_i = \tau$ all i).

Proposition 4.2: If $A < 0$, the origin is the only equilibrium point. If $(\Lambda|A|)^{1/N}\tau \cdot \cos(\pi/N) < 1$, the origin is stable, and if $(\Lambda|A|)^{1/N}\tau\cos(\pi/N) > 1$, the origin is unstable.

Proof: The equation for the equilibrium is obtained by setting the right hand side of (4.4) equal to zero. We get

$$u_1 = \tau w_{1N} f_{\lambda_N}(\tau w_{N,N-1} f_{\lambda_{N-1}}(...\tau w_{21} f_{\lambda_1}(u_1))). \tag{4.5}$$

Since sign$(f(u))$=sign(u), the sign of the right hand side of (4.5) is $-$sign(u_1), since there is an odd number of negative $w_{i,i-1}$'s. Thus, the only solution is $u_1 = 0$, and hence $u_i = 0$, all i, and the origin is the only equilibrium point. The stability of the origin could be determined by linearizing the system around the origin, to obtain

$$\frac{d\mathbf{u}}{dt} = \mathbf{J}(0)\mathbf{u},$$

where $\mathbf{J}(0)$ is the Jacobian matrix, its $(i,j)^{th}$ element being given by $\frac{\partial F_i}{\partial u_j}$. The eigenvalues β_k of the Jacobian are evaluated as:

$$\beta_k = -\frac{1}{\tau} + (\Lambda|A|)^{1/N} e^{j\pi(2k-1)/N} \qquad k = 1, ..., N.$$

If $(\Lambda|A|)^{1/N}\tau\cos(\pi/N) < 1$, all eigenvalues have negative real parts, and hence the origin in the non-linear system (4.4) is stable. If $(\Lambda|A|)^{1/N}\tau\cos(\pi/N) > 1$, there are eigenvalues with positive real parts, and hence the origin in (4.4) will be unstable. ■

As a result of Proposition 4.2, the origin is stable at small gains or small weights. As we increase the gains or the weights, at the point when $(\Lambda|A|)^{1/N}\tau \cdot \cos(\pi/N) = 1$, a Hopf bifurcation occurs. The Jacobian has a pair of pure imaginary eigenvalues. The origin loses stability, and a stable periodic orbit is created. We prove the following:

Theorem 4.1: If $(\Lambda|A|)^{1/N}\tau\cos(\pi/N) > 1$, then there exists a periodic solution, and all trajectories converge to either the unstable equilibrium at the origin, or to a periodic orbit.

Proof: We will use a theorem by Mallet-Paret and Smith [28], which proves several results on monotone cyclic feedback systems. Define a new set of variables z_i by

$$u_i = w_{i,i-1}(z_{i-1} - \tau_i), \qquad i = 1, ..., N$$

with, as usual, $z_0 = z_N$ and $w_{10} = w_{1N}$. The system (4.4) becomes

$$\frac{dz_i}{dt} = -\frac{z_i}{\tau_{i+1}} + f_{\lambda_i}\left[w_{i,i-1}(z_{i-1} - \tau_i)\right] + 1 \equiv G_i(z_i, z_{i-1}), \qquad i = 1, ..., N. \quad (4.6)$$

For this system the orthant $\mathbf{R}_+^N = [0, \infty)^N$ is invariant. Moreover, it has a unique equilibrium, using Proposition 4.2. It is straightforward to verify that it is a monotone cyclic feedback system; i.e., there are α_i's 1 or -1 such that

$$\alpha_i \frac{\partial G_i}{\partial z_{i-1}} > 0 \qquad \text{all} \;\; z_i, \; z_{i-1},$$

and G_i is C_{N-1}. Also, one can prove that every trajectory is bounded, as one can construct a cube $[0, U]^N$, large enough such that the vector field is pointing inward. Also, the Jacobian has at least two eigenvalues with a positive real part, if $(\Lambda|A|)^{1/N}\tau\cos(\pi/N) > 1$. Using Theorem 4.1 in Mallet-Paret and Smith, we can conclude that the system (4.6), and hence our original system (4.4) have a periodic solution, and that all trajectories converge to either the unique equilibrium, or to a periodic orbit. ∎

The implication of Theorem 4.1 is that for the ring architecture there is no complicated behavior such as quasi-periodic or chaotic motion. Also, because the unique equilibrium is unstable, almost all trajectories converge to a periodic orbit.

4.6 LAYERED NETWORKS

Now, consider the general case of layered networks. The layers are connected in a circular fashion, as shown in Figure 4.2. Within each layer there are no lateral connections. Let $\mathbf{W}^{l,l-1}$ be the matrix of weights from layer $l-1$ to layer l, and let L be the number of layers. Assume that the weights from one layer to the next are of similar sign. By Hirsch's Theorem, an even number of inhibitory layers result in convergent behavior. So again we will consider only the case of an odd number of inhibitory layers. If the weights from one layer to the next are equal, and the time constants and the gains of each layer are equal, then the layered network behaves effectively as a ring. This is proven in the following theorem.

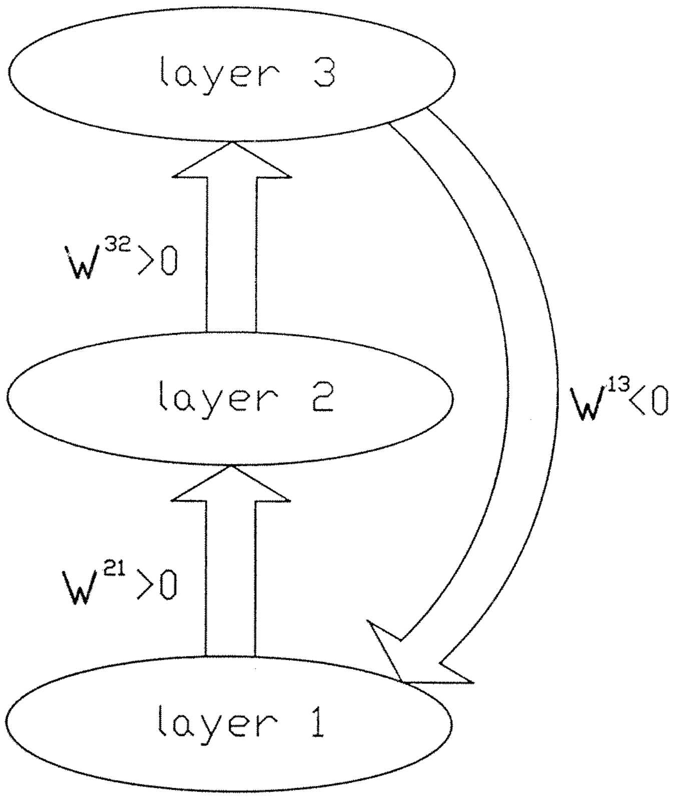

Figure 4.2: A three layer network: the weights from layer 1 to layer 2 and from layer 2 to layer 3 are positive, and those from layer 3 to layer 1 are negative. Provided the gains are high enough, for most possible choices of the weights, this structure

oscillates.

Theorem 4.2: Let the time constants and the gains of layer l be $\tau(l)$ and $\lambda(l)$ respectively, and let $\left[\mathbf{W}^{l,l-1}\right]_{(i,j)} = w(l)$. Then, for any initial conditions, the outputs of any two neurons in the same layer differ by a quantity that decreases to zero exponentially fast. The limiting behavior of the network corresponds to that of a ring with $\tau_i = \tau(l)$, $\lambda_i = \lambda(l)$ and $w_{i,i-1} = w(l)m(l-1)$, where $m(l-1)$ is the number of neurons in layer $l-1$ (modulo L).

Proof: Let u_{i_1} and u_{i_2} be the outputs of two neurons in the same layer l. Then,

$$\frac{du_{i_1}}{dt} - \frac{du_{i_2}}{dt} = -\frac{u_{i_1} - u_{i_2}}{\tau(l)},$$

and thus $u_{i_1} - u_{i_2} \longrightarrow 0$ exponentially fast, hence proving the first part.

Let $u(l)$ be the asymptotic output of a neuron in layer l. Then,

$$\frac{du(l)}{dt} = -\frac{u(l)}{\tau(l)} + w(l)m(l-1)f_{\lambda(l-1)}(u(l-1)),$$

which proves the second part of the theorem. ∎

Theorem 4.2 implies, of course, that if the layered network oscillates, then the oscillations of each layer zero phase lock.

We found by simulations that zero phase locking is still preserved if we relax some of the conditions of Theorem 4.2, as follows. We have generated the weights of each $\mathbf{W}^{l,l-1}$ randomly, but with keeping the signs fixed (the time constants of each layer are constant). We have found the remarkable result that with an odd number of inhibitory layers, 0 phase locking of each layer still holds exactly (see Figure 4.3 for a three-layer example). The peaks of the oscillations of the neurons of each layer coincide exactly, as do the zero crossings also. Although we have obtained this result in all of tens of simulation experiments we have performed,

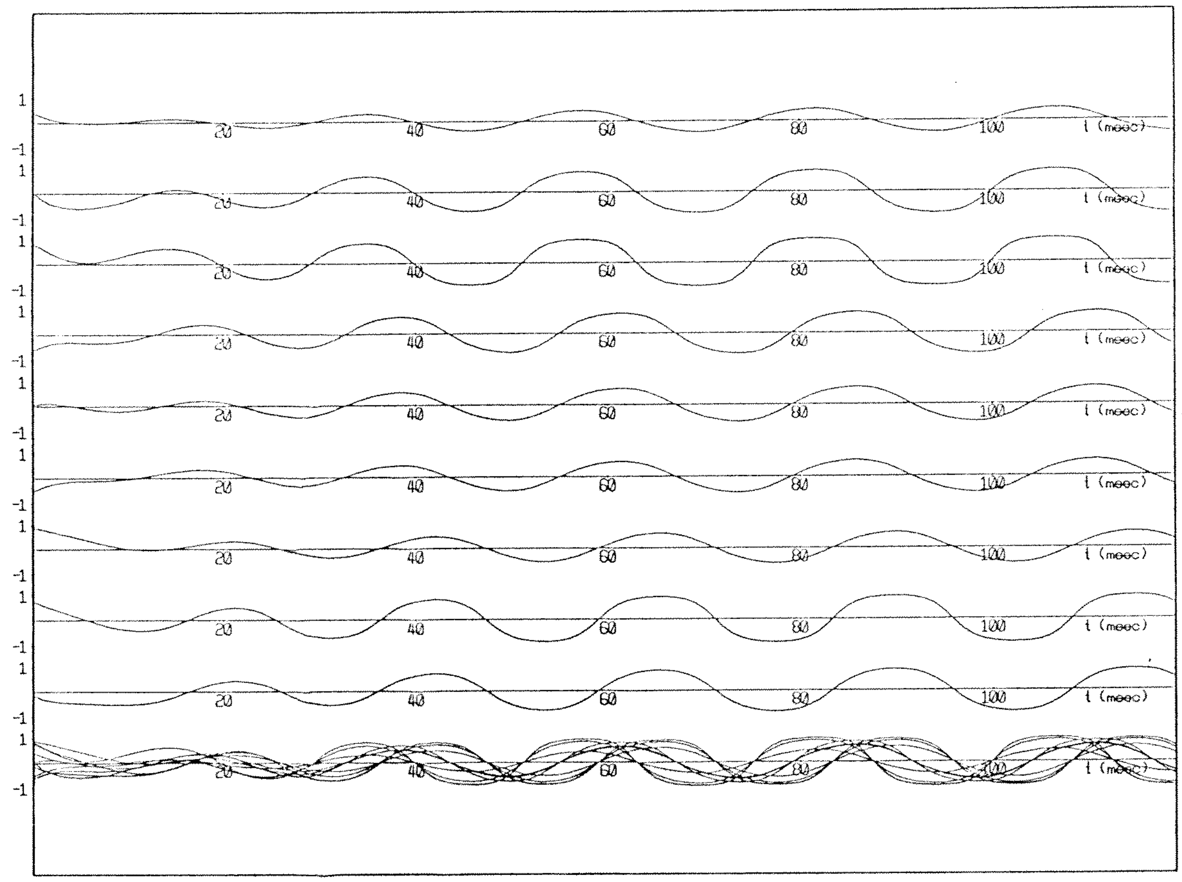

Figure 4.3: The oscillations (in the x-space) of the three-layer network of Figure 4.2 with three neurons per layer. The upper three, middle three, and lower three curves correspond to layer 1, layer 2, and layer 3, respectively. For phase comparison purposes, the lowest graph superimposes the outputs of all 9 neurons. The weight matrices are generated randomly, and the signs of the weights are as given in Figure 4.2. We observe that the oscillations of each layer zero phase lock.

it does not mean that any network with an odd number of inhibitory layers and equal time constants for each layer will result in 0 phase locking. One can easily find cases where phase locking does not necessarily occur; for instance, it is easy, with the proper connections, to imbed a long ring with an even number of negative connections within such a layered network. We have also found by simulations that the phase locking remains very robust with respect to perturbations in the time constants of each layer. This means that varying the time constants within each layer will result in only a slight change in the relative phases of the neurons of each layer. The appendix presents a proof of phase locking for some cases of random weight matrices.

We have mentioned in the introduction that the labeling hypothesis is based on computations involving the phases, the frequency, and/or the amplitudes of the oscillations. It is therefore important to derive their relationships with the parameters of the model (weights, gains, and time constants). Again, consider a network with an odd number of inhibitory layers. From simulations we have observed that the oscillations in the u-space are close to sinusoids. Let the output of the i^{th} neuron of layer l be approximated as $u_i(l) = A_i(l)\sin(\omega t + \phi_l)$, where ω is the angular frequency of the oscillation, and ϕ_l is the phase of the oscillations of layer l. Assume that we are close to the Hopf bifurcation point; i.e., the oscillations are of small amplitudes. Using the approximation

$$f_{\lambda_i}\left[u_i(l)\right] = f_{\lambda_i}\left[A_i(l)\sin(\omega t + \phi_l)\right] \approx \lambda_i A_i(l)\sin(\omega t + \phi_l),$$

we obtain upon substitution in the differential equation set the phase shift $\phi_{l,l-1}$ between the oscillations of layers l and $l-1$:

$$\phi_{l,l-1} = \begin{cases} -\arctan\left[\tau(l)\omega\right] & \text{if } \mathbf{W}^{l,l-1} > 0, \\ \pi - \arctan\left[\tau(l)\omega\right] & \text{if } \mathbf{W}^{l,l-1} < 0. \end{cases} \tag{4.7}$$

By going through all phase shifts once around the loop, we obtain the following expression for the evaluation of the frequency of oscillation,

$$\sum_{l=1}^{N} \arctan\left[\tau(l)\omega\right] = \pi. \tag{4.8}$$

The three layer case ($N = 3$) is of particular interest. It can be shown that

$$\omega = \sqrt{\frac{\tau_1 + \tau_2 + \tau_3}{\tau_1 \tau_2 \tau_3}},$$

which, for the case of equal time constants ($= \tau$), gives the period $T = 2\pi/\omega = 2\pi\tau/\sqrt{3} \approx 3.63\tau$, and $\phi_{l,l-1} = \pi/3$.

When the gains and/or the weights are large, the oscillations have a large amplitude, and an expression for the frequency and phase shifts can also be derived. We can approximate the oscillations in the x-space as a square wave: $f_{\lambda_i}\left[u_i(l)\right] \approx \text{sign}\left[\sin(\omega t + \phi_l)\right]$. It is easy to derive the phase shifts, as an expression for the solution of the differential equation set can be obtained. We get

$$\phi_{l,l-1} = \begin{cases} -\omega\tau(l)\ln\left[2/(1 + e^{-\pi/\omega\tau(l)})\right] & \text{if } \mathbf{W}^{l,l-1} > 0, \\ \pi - \omega\tau(l)\ln\left[2/(1 + e^{-\pi/\omega\tau(l)})\right] & \text{if } \mathbf{W}^{l,l-1} < 0. \end{cases} \tag{4.9}$$

Summing the phase shifts around the loop, we get for the frequency of oscillation

$$\sum_{l=1}^{N} \omega\tau(l)\ln\left[2/(1 + e^{-\pi/\omega\tau(l)})\right] = \pi. \tag{4.10}$$

Although in the derivation of (4.7) and (4.8) we have assumed that we are close to the bifurcation point, simulations show that the expressions remain reasonably accurate beyond the bifurcation. Moreover, in general the high gain expressions (4.9) and (4.10) result in values not too different from the values given in (4.7) and (4.8), respectively. Intermediate gain or weight values result in frequencies and phase shifts in between those given by (4.7), (4.8) and (4.9), (4.10), respectively (see Table 4.1 and Figure 4.4 for simulation results). We therefore observe that the frequency and the phase shifts are to some extent not much affected by the weight values.

λ	0.3	1	2	3	4	5
T (msec)	18.5	16.1	15.1	14.8	14.7	14.6
ϕ_{12} (degrees)	69.9	71.4	70.9	70.7	70.3	70.3
ϕ_{23} (degrees)	76.7	76.3	76.3	76.5	76.7	76.7

Table 4.1: The period and the phase shifts of the three-layer network of Figure 2. The weights are generated randomly and the time constants of layers 1, 2, and 3 are 2, 8, and 12 msec, respectively. The theoretical estimates for the period and the phase shifts for the weakly non-linear range using Equations (4.7) and (4.8) are: $T = 18.6$ msec, $\phi_{12} = 69.8^\circ$, and $\phi_{23} = 76.2^\circ$, thus agreeing to some extent with the measured values for small λ. The theoretical estimates for the high gain limit, using Equations (4.9) and (4.10), can also be obtained as: $T = 14.5$ msec, $\phi_{12} = 70.3^\circ$, and $\phi_{23} = 76.6^\circ$. We can see that they also agree with the measured values for high λ.

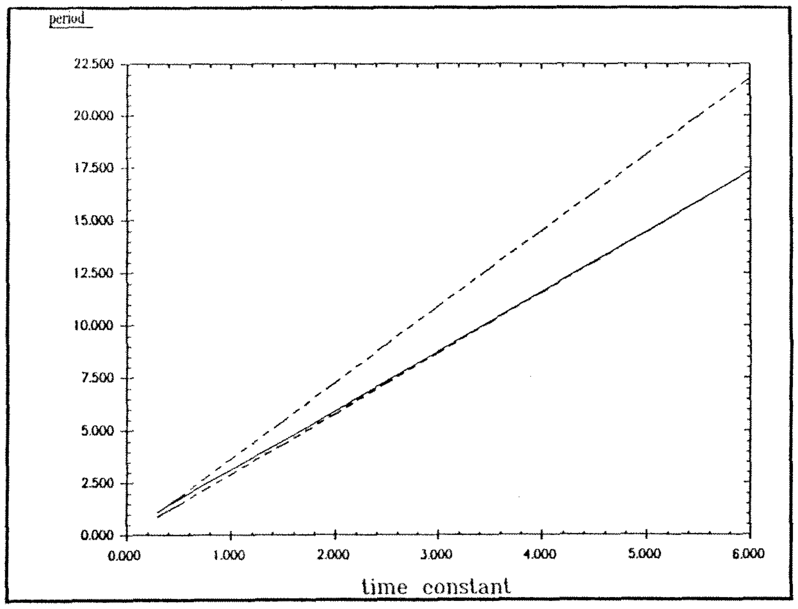

Figure 4.4: The period in a three-layer network with randomly generated weights, with equal time constant τ for all neurons (keeping λ fixed). The relation is close to a straight line and agrees with the theoretically calculated values of 3.63τ and 2.89τ (shown as the two dashed lines) for small τ and large τ, respectively.

4.7 PHASE LOCKING MECHANISM

We present here a mechanism for the synchronization of two remote non-connected populations of neurons. This gives a tentative explanation to the results of Gray *et al.* It also sheds some light on how the phases of aggregates of neurons can be adjusted, an essential step for the labeling process. Consider two non-connected similar neural network oscillators, such as the layered type considered in the last section. Let the two networks be initially out of phase. If an input in the form of a strong short pulse is applied simultaneously to one or more layers in both networks, then the two networks rapidly synchronize (see Figures 4.5 and 4.6 for an illustration). The explanation is as follows. Because of the pulse, each network loses its oscillation, and converges to an identical equilibrium at one of the corners of the hyper-cube (in x-space). When the pulse ends, both networks relax together to the limit cycle, and the oscillations of both networks become in phase. This mechanism is robust with respect to small differences in the weights of the two networks, because in any case the equilibrium is at approximately the same position at the corner of the hypercube. The only difference is the path back to the limit cycle, which does not affect the phase much. We remark that this scheme is analogous to the case of two initially out of phase pendula, given a common strong impulse in a specific direction, which results in restoring their synchronization. It is important to note that the applied pulse need not represent only an external stimulus. It could also represent a feedback signal from higher centers of the brain. For example, a possible scenario is that the largest effect on phase locking could happen when an external input together with a feedback pulse from higher centers arrive simultaneously.

4.8 NETWORKS WITH LATERAL CONNECTIONS

Consider the usual network with an odd number of inhibitory layers. If we add

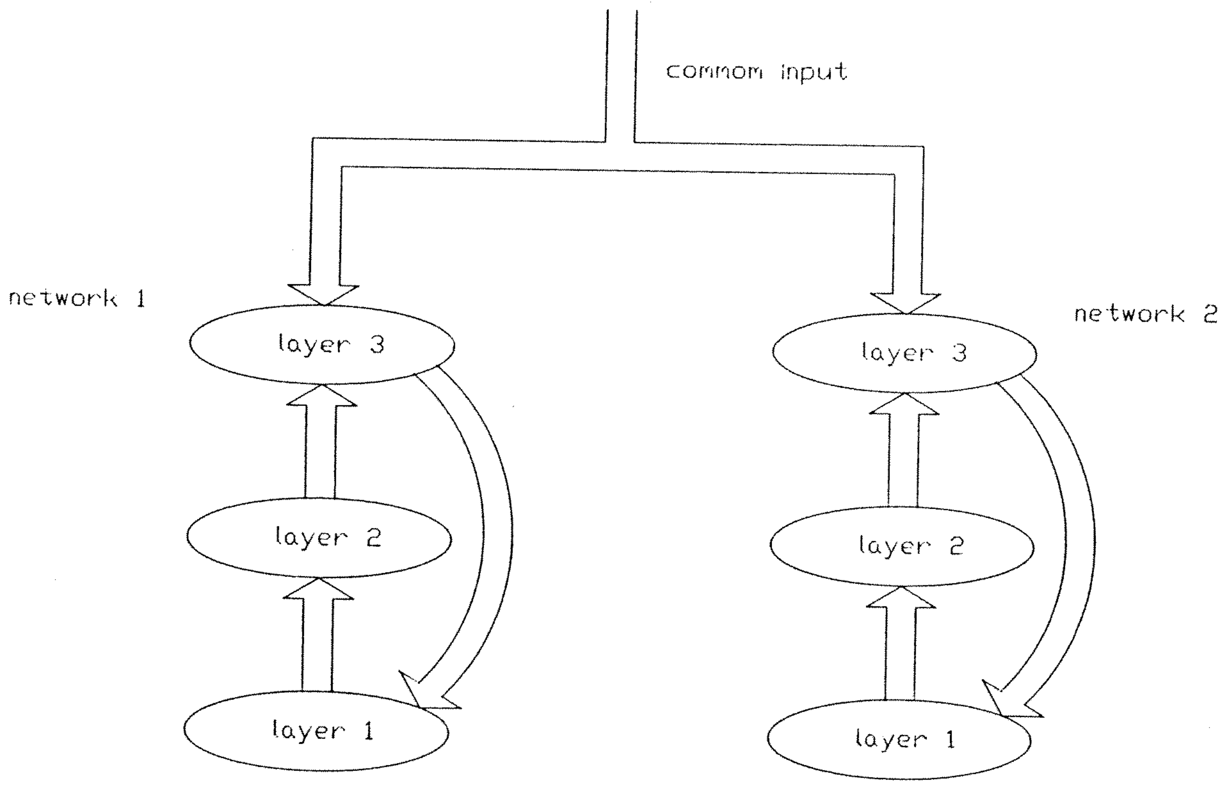

Figure 4.5: A phase locking mechanism: A strong common input applied to two non-connected populations of neurons results in the 0 phase locking of the oscillations of both populations.

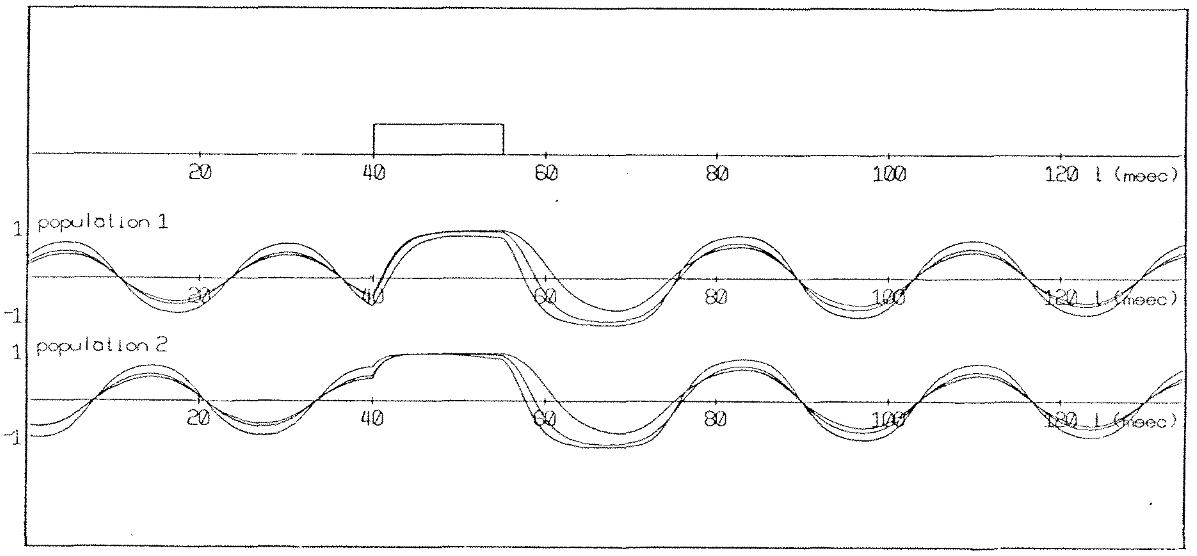

Figure 4.6: The phase locking mechanism between two non-connected populations of neurons. The upper curves represent the outputs of the first layer in a three-layer network with three neurons per layer. The lower curves represent the outputs of a similar network, but initially out of phase from the other network. A pulse applied simultaneously to the first layer of each of the two networks results in synchronizing the oscillations of the corresponding layers of the two networks.

lateral connections, i.e., connections within each layer, but no self-connections, then the following is observed from computer simulations. Provided that the magnitudes of the lateral connection weights are not too large compared to the magnitudes of the inter-layer weights, this architecture oscillates. If the signs of the lateral connection weights of each layer are constant, then the oscillations are approximately in phase. For instance for a three-layer network the oscillations are found to be in phase $\pm 2 - 4\%$ if the lateral connection weights are positive and comparable in magnitude to the inter-layer weights. If the signs of the lateral connection weights are negative, then the robustness of the oscillation and the phase locking becomes worse to some extent. If the signs of the lateral connection strengths are not constant, then phase locking almost does not hold. Of course these observations hold in general when the weights are randomly generated, and do not necessarily hold for every possible choice of the weights.

4.9 THE SPIKING MODEL

In the spiking model, if the neuron potential u_i is below a threshold value θ_i, then the i^{th} neuron integrates according to the charging equation

$$\frac{du_i}{dt} = -\frac{u_i}{\tau_i} + \sum_{j=1}^{N} \sum_{t_j} w_{ij} e^{-(t-t_j)/\tau}, \tag{4.11}$$

where the form of the excitatory or the inhibitory post-synaptic potential is modeled by the decaying exponential $e^{-t/\tau}$ of width about τ, and the t_j's are the firing times of neuron j up to time t (in the simulations we have taken the intervals $t - t_j$ to be restricted to a window $(0, w)$). If the activity u_i reaches the threshold value θ_i at time t, then t is a firing instant for neuron i, u_i is reset to 0, and neuron i is prevented from firing during a refractory period r_i.

We will concentrate here directly on the most interesting case of layered networks. In our model the thresholds need to be scaled. We chose $\theta_i = f_i \sum_j w_{ij}$, $f_i >$

0 where f_i is equal for all neurons belonging to the same layer. Consider the i^{th} neuron with $u_i(0) = c$, starting to charge at time 0 by summing the effects of the incoming synaptic potentials present in its past. Then (4.11) can be rewritten as

$$\frac{du_i}{dt} = -\frac{u_i}{\tau_i} + Ae^{-t/\tau} \tag{4.12}$$

with $A = \sum_j \sum_{t_j} w_{ij} e^{t_j/\tau}$. Integration of (4.12) results in the following expression for the neuron potential,

$$u_i(t) = \begin{cases} (At + c)e^{-t/\tau} & \text{if } \tau_i = \tau \\ A\frac{\tau_i \tau}{\tau_i - \tau}(e^{-t/\tau_i} - e^{-t/\tau}) + ce^{-t/\tau_i} & \text{if } \tau_i \neq \tau. \end{cases} \tag{4.13}$$

If a new spike occurs at time t_k in a neuron connected to neuron i while it is charging, we can still use Equation (4.13) with the proper adjustments (i.e., we use (4.13) as it is till time t_k, then reset the time axis, A and c using $u(t_k) = c$). In any case, the main difficulty arises from the fact that the equation $u_i(t) = \theta_i$ cannot be solved analytically to determine the firing times of the neurons. Rescigno *et al.* [29] have analyzed the behavior of one such element under a sinusoidal input. They showed that for all values of the parameters, there exists a pattern of firing times which repeats itself periodically after a finite number of pulses in the absence of noise, and this pattern is stable against perturbations. Even the precise response of a neuron described by (4.11) to a periodic train of spikes (i.e. $t_j = j2\pi/\omega$) is not completely known when the window of integration is taken to be infinite ($w = \infty$). However, if we assume (and that will be the case in all that follows) that a unit has only a short memory, in the sense that it integrates only those incoming spikes that have occurred since the last time it fired, then it is obvious that such a neuron, when stimulated by a periodic train of spikes, will fire periodically and at constant phase with respect to the train.

In order to avoid negative thresholds, all the weights between a layer and the next are always taken to be non-negative. Typically, two types of stable behavior are commonly observed: The activity of the network dies out or a periodic pattern of firings establishes itself (see Figure 4.7 for a three-layer example). If the threshold values are too high or if the number of initial spikes is too small, then the activity of the network dies out because the neuron potentials u_i cannot reach the threshold θ_i. If the thresholds are not too high and the initial condition is that all the neurons in the first layer fire at time $t = 0$, then all the neurons in the second layer will also fire simultaneously, even in the case of random synaptic weights. The reason for that is that the firing time t of a neuron in the second layer is the solution of (consider the case $\tau_i \neq \tau$)

$$\theta_i = f_i \sum_j w_{ij} = \sum_j w_{ij} \frac{\tau_i \tau}{\tau_i - \tau}(e^{-t/\tau_i} - e^{-t/\tau}). \tag{4.14}$$

The synaptic weights cancel from both sides, and if we assume the time constants and the fractional thresholds f_i to be constant within a layer, the time t of firing is independent of the unit considered. Obviously, a similar analysis holds for the following layers, leading to a stable periodic pattern of activity where all neurons in each layer fire synchronously, one layer after the other.

The maximal value of the function on the right hand side of (4.13) is reached when $t = t_{max} \approx \tau\tau_i(\ln\tau_i - \ln\tau)/(\tau_i - \tau)$. So, to fix these ideas, consider for simplicity a three-layer network where all the units have the same parameter values $\tau_i = 8$ msec, $\tau = 6$ msec, $f_i = f$ and a refractory period $r = 4$ msec. Then $t_{max} \approx 6.9$ msec, corresponding to $f = f_{max} \approx 2.5$, and for $t = r = 4$ msec, $f = f_{ref} \approx 2.2$. As a result, there are three possible ranges of regimes depending on the value of the common threshold parameter f in the network. If $0 < f < 2.2$, then the threshold is reached before the end of the refractory period. The period of the network is therefore given by $T = 3r = 12$ msec, corresponding to a frequency of 83 Hz. If $2.2 < f < 2.5$, then the threshold is reached at a time $4 < t < 6.9$ obtained by solving (4.14). This yields a period $12 < T < 20.7$ corresponding

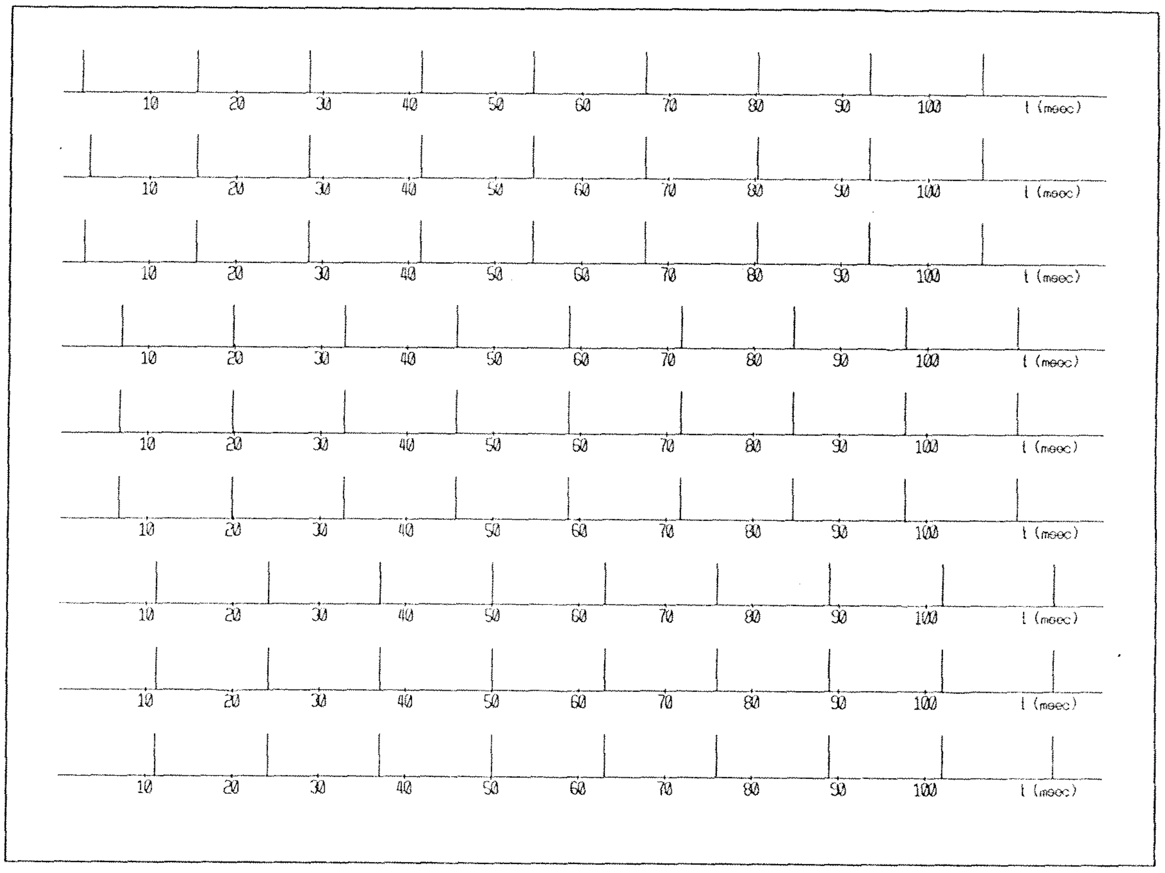

Figure 4.7: The outputs of a three-layer spiking model with three neurons per layer. The threshold of each neuron is chosen as 2.3 times the sum of weights converging onto it, $\tau_i = 6$ msec for all i, and $\tau = 8$ msec. The upper three, middle three, and lower three graphs represent the outputs of layer 1, layer 2, and layer 3, respectively. The bars represent the spikes. The first layer is given an external stimulus so that all its neurons fire near time zero. The dispersion of the firing instants of the following layers decreases progressively until the spikes of each layer are simultaneous.

to frequencies in the range 48-83 Hz. Finally, if $2.5 < f$, then the potentials never reach the threshold and the activity dies out. The case of L layers, with $L = 2$, or $L > 3$ can be analyzed similarly. In any case, the period is always entirely determined by the relative thresholds f_i, the time constants, and possibly the refractory periods. The connection weights appear only through their sum at each neuron, which in random networks can be assumed to be roughly constant. As in the continuous case, the details of the connections are irrelevant. Notice also that two similar, but independent layered networks that are activated simultaneously in two corresponding layers, will trivially synchronize their periodic activity.

An important feature of the previous analysis is that under similar conditions, the periodic behavior observed is unchanged if we add lateral intra-layer connections or if the inter-layer connectivity is local. Both analytically and in simulations it is easy to see that the behavior is also robust with respect to several other perturbations of the initial conditions:

- when only a (large) fraction of the cells in one layer is initially activated;
- when a low level of possibly random activity exists in the network prior to the activation of the cells in one layer; notice in particular that isolated random firing is insufficient in general to activate an entire layer;
- when the initial activation of the cells in one layer is not exactly simultaneous but occurs over a short interval of time; in the case of random connectivity, it is easy to see that the firing times in the following layer will be concentrated around their mean with an even smaller dispersion (see Figure 4.7).

4.10 CONCLUSION

In the present work we have studied oscillations and synchronization in neural networks. We have shown that a layered network possesses oscillation and zero phase locking properties, which are very robust with respect to the weight values. A mechanism for the synchronization of two non-connected populations of neurons using a common input is proposed. This mechanism is simple and general enough to apply to a large variety of architectures and models of neural oscillators. In the scheme we have proposed, synchronization of the remote neural populations is through a strong input and/or a feedback signal from higher brain centers. A point that has to be investigated further is whether through using the continuous time model (4.1) can lateral connections between populations of neurons have an additional role in propagating synchronization upon the arrival of a common input. An important issue for further research is to develop schemes where the frequency and amplitude of the oscillations also play a role in the labeling process. The frequency might, for example, be a measure for the speed of a moving object, while the amplitude of the oscillation could indicate a measure of the distance of the object. On first thought, these problems seem to be hard, since a change in the neuron weights on a fast time scale seems to be required. However, a fast adaptation of the weights in the brain is not ruled out, and in fact several researchers hypothesized its existence (see von der Malsburg [11]). An alternative is to consider a high order network model, where a neuron output can serve as a gate for controlling the effect of the other neurons. This would adjust the coupling strengths of a number of neural oscillators, and thus would provide an effective mechanism for controlling the phase interactions, possibly using control signals from higher brain centers.

Certainly, more experimental results are awaited, so that we can correctly direct the theoretical investigation, which, we hope, will shed light on the problem of the extraction of high level features.

APPENDIX: A PHASE LOCKING CONDITION

We prove here a phase locking condition for networks with random weight matrices. Consider a multi-layer network with $\tau_i = \tau$ and $\lambda_i = \lambda$ for all neurons i. Assume that the entries in $\mathbf{W}^{(l,l-1)}$ are random but of constant sign, with an odd number of inhibitory layers. Assume that the gain and/or the weights are small enough such that the system is close to the Hopf bifurcation point; i.e., we have small oscillations. The eigenvectors of the Jacobian $\mathbf{J}(0)$ at the bifurcation corresponding to the pair of pure, imaginary eigenvalues give the approximate phases of the oscillations. This can be seen by plugging $\mathbf{a}e^{j\omega}$ in the differential equations, and retaining only linear terms in the Taylor series expansion ($\mathbf{a}$ represents the complex vector of phases of the oscillations $(e^{j\phi_1}, ..., e^{j\phi_L})^T$). Let the eigenvector of the Jacobian corresponding to eigenvalue $j\omega$ be of the form $\mathbf{v}^T = (\mathbf{v}_1^T, ..., \mathbf{v}_L^T)^T$, where $\mathbf{v}_l$ is the component associated with layer l. Oscillations will occur with approximate zero phase difference within each layer if $\mathbf{v}_l$ is of the form $\mathbf{v}_l = c_l \mathbf{z}_l$, where c_l is a complex constant and $\mathbf{z}_l$ is a vector with strictly positive components. Since $\mathbf{v}$ is an eigenvector (let the eigenvalue be β), it must satisfy the relations

$$\lambda \mathbf{W}^{l,l-1}\mathbf{v}_{l-1} = (\frac{1}{\tau} + \beta)\mathbf{v}_l, \qquad l = 1, ..., L.$$

Hence,

$$\mathbf{W}^{1,L}...\mathbf{W}^{2,1}\mathbf{v}_1 = (\frac{1}{\tau} + \beta)^L \lambda^{-L}\mathbf{v}_1$$

$$...$$

$$\mathbf{W}^{L,L-1}...\mathbf{W}^{1,L}\mathbf{v}_L = (\frac{1}{\tau} + \beta)^L \lambda^{-L}\mathbf{v}_L.$$

Therefore, the matrices $\mathbf{S}_l \equiv \mathbf{W}^{l,l-1}\mathbf{W}^{l-1,l-2}...\mathbf{W}^{l+1,l}$, $l = 1, ..., L$ have the same set of eigenvalues. If β is an eigenvalue of $\mathbf{J}(0)$, then $(\frac{1}{\tau} + \beta)^L\lambda^{-L}$ is an eigenvalue of $\mathbf{S}_1, ..., \mathbf{S}_L$. Conversely, it can be checked that if γ is an eigenvalue of $\mathbf{S}_1$, then all the complex numbers of the form $-1/\tau + \lambda\gamma^{1/L}$ ($\gamma^{1/L}$ is any L-th root of γ) are eigenvalues of $\mathbf{J}(0)$.

Let $\rho(\mathbf{M})$ denote the spectral radius of a matrix $\mathbf{M}$ (i.e., the maximum of the magnitudes of the eigenvalues). The Perron-Frobenius theorem (see Gantmacher [30]) states that if $\mathbf{M}$ is with non-negative components and irreducible (irreducible means that for any i and k, there is a sequence $j_1, ..., j_p$, such that $m_{ij_1}, m_{j_1 j_2}, ..., m_{j_p k}$ are nonzero), then $\mathbf{M}$ has a positive real eigenvalue equal to its spectral radius $\rho(\mathbf{M})$. This eigenvalue is simple and has an eigenvector with positive components. The matrices $\mathbf{S}_1, ..., \mathbf{S}_L$ are negative and almost surely irreducible. By the Perron-Frobenius theorem, $\mathbf{S}_1$ has a simple eigenvalue γ^* with $\gamma^* = -\rho = -\rho(\mathbf{S}_1)$, the spectral radius of $\mathbf{S}_1$ with eigenvector $\mathbf{v}_1^*$ of real and positive components. The complex numbers

$$\beta_k = -\frac{1}{\tau} + \lambda \rho^{1/L} e^{j\pi(2k+1)/L}, \qquad k = 0, ..., L-1$$

are eigenvalues of $\mathbf{J}(0)$ with eigenvector $\mathbf{z}^k = (\mathbf{z}_1^k, ..., \mathbf{z}_L^k)$ and

$$\mathbf{z}_l^k = \left(\frac{\lambda}{\frac{1}{\tau} + \beta_k}\right)^{l-1} \mathbf{W}^{l,l-1}\mathbf{W}^{l-1,l-2}...\mathbf{W}^{2,1}(\mathbf{v}_1^*) \qquad j = 2, ..., L$$

and $\mathbf{z}_1^k = \mathbf{v}_1^*$. The rightmost eigenvalue of $\mathbf{J}(0)$ is the one eventually associated with the bifurcation. Therefore, if β^* is the rightmost eigenvalue, then oscillations occur with (approximate) 0 phase difference within each layer. This happens if

$$\text{Re}(\beta^* = -\frac{1}{\tau} + \lambda \rho^{1/L} \cos(\pi/L) > \text{Re}(\alpha)$$

for any other eigenvalue α of $\mathbf{J}(0)$. Hence, if δ is the eigenvalue of $\mathbf{S}_1$ with the second largest magnitude, the condition

$$|\delta| < \rho 2^{-L} \leq \rho\left(\cos\frac{\pi}{L}\right)^L$$

guarantees phase locking. There are simple situations where the previous condition is satisfied. Assume for example that $m_1 = ... = m_L \equiv M$ and that the weight matrices $\mathbf{W}^{l,l-1}$ are symmetric (which is true if, for instance, the connection weight between any two neurons depends only on their inter-distance). Assuming that the entries in $\mathbf{S}_l$ are independent identically distributed for $i < j$; then it is known that there is a very sharp drop between the size of the first and the second eigenvalue (refer to Juhász [31] and Füredi and Komlós [32]). The first eigenvalue is of order M, whereas the second eigenvalue is of order $\sqrt{M}$.

REFERENCES

[1] M. Hirsch, "Convergent activation dynamics in continuous time networks," *Neural Networks*, Vol. 2, No. 5, pp. 331-350, 1989.

[2] W. Freeman and C. Skarda, "Spatial EEG patterns, non-linear dynamics and perception: the Neo-sherringtonian view," *Brain Res. Rev.*, Vol. 10, pp. 147-175, 1985.

[3] W. Freeman and B. van Dijk, "Spatial patterns of visual cortical fast EEG during conditioned response in a rhesus monkey," *Brain Res.*, Vol. 422, pp. 267-276, 1987.

[4] E. Pöppel and N. Logothetis, "Neuronal oscillations in the human brain," *Naturwissenschaften*, Vol. 73, pp. 267-268, 1986.

[5] T. Pavlidis, *Biological Oscillators: Their Mathematical Analysis*, Academic Press, 1973.

[6] C. Gray, P. König, A. Engel, and W. Singer, "Oscillatory responses in cat visual cortex exhibit inter-columnar synchronization which reflects global stimulus properties," *Nature*, Vol. 338, No. 6213, pp. 334-337, 1989.

[7] C. Gray and W. Singer, "Stimulus-specific neuronal oscillations in orientation

columns of cat visual cortex," *Proc. Nat. Acad. Sci.*, Vol. 86, pp. 1698-1702, 1989.

[8] R. Eckhorn, R. Bauer, W. Jordan, M. Brosch, W. Kruse, M. Munk, and H. Reitboek, "Coherent oscillations: a mechanism for feature linking in the visual cortex?" *Biol. Cybernetics*, Vol. 60, pp. 121-130, 1988.

[9] S. Kuffler, J. Nichols, and A. Martin, *From Neuron to Brain*, Sinauer Associates, Sunderland, MA, 1984.

[10] C. von der Malsburg, "The correlation theory of brain functions," Internal Report 81-2, Department of Neurobiology, Max Plank Institute for Biophysical Chemistry, 1981.

[11] C. von der Malsburg, "Nervous structures with dynamical links," *Ber. Bunsenges. Phys. Chem.*, Vol. 89, pp. 703-710, 1985.

[12] C. von der Malsburg and W. Schneider, "A neural cocktail-party processor," *Biol. Cybernetics*, Vol. 54, pp. 29-40, 1986.

[13] M. Abeles, *Local Cortical Circuits: Studies of Brain Functions*, Springer, New York, Vol. 6, 1982.

[14] F. Crick, "Function of the thalamic reticular complex: the searchlight hypothesis," *Proc. Nat. Acad. Sci.*, Vol. 81, 4586-4590, 1984.

[15] P. Baldi, J. Buhmann, and R. Meir, "Computing with arrays of coupled oscillators: two dimensional effects," submitted for publication.

[16] P. Baldi and A. Atiya, "Oscillations and synchronization in neural networks: an exploration of the labeling hypothesis," *International Journal of Neural Systems*, Vol. 1, No. 2, pp. 103-124, 1990.

[17] A. Hodgkin and A. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *Journal of Physiology*, Vol. 117, pp. 500-544, 1952.

[18] J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Nat. Acad. Sci*, Vol. 81, pp. 3088-3092, 1984.

[19] L. Partridge, "A possible source of nerve signal distortion arising in pulse rate encoding of signals," *Journal of Theoretical Biology*, Vol. 11, pp. 257-281, 1966.

[20] B. Knight, "Dynamics of encoding in a population of neurons," *The Journal of General Physiology*, Vol. 59, pp. 734-766, 1972.

[21] S. Grossberg and M. Cohen, "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks," *Trans. Syst., Man, Cybernetics*, Vol. SMC-13, pp. 815-826, 1983.

[22] M. Hirsch, "Convergence in neural networks," *Proc. 1987 Int. Conf. on Neural Net.*, San Diego, 1987.

[23] B. Baird, "Bifurcation analysis of oscillating network model of pattern recognition in the rabbit olfactory bulb," *Neural Networks for Computing*, J. Denker,*Ed.*, Snowbird, Utah, AIP Conf. Proceedings, 1986.

[24] B. Baird, "Bifurcation theory methods for programming static or periodic attractors and their bifurcations in dynamic neural networks," *Proc. of the IEEE Int. Conf. on Neural Networks*, San Diego, California, 1988.

[25] Z. Li and J. Hopfield, "Modeling the olfactory bulb and its neural oscillatory processings," *Biol. Cybernetics*, Vol. 61, No. 5, pp. 379-392, 1989.

[26] B. Hassard, N. Kazarinoff, and Y.-H. Wan, *Theory and Applications of Hopf Bifurcation*, Cambridge University Press, Cambridge, 1981.

[27] A. Nayfeh and D. Mook, *Nonlinear oscillations*, Wiley, New York, 1979.

[28] J. Mallet-Paret and H. Smith, "The Poincaré-Bendixson theorem for monotone cyclic feedback systems," submitted for publication.

[29] A. Rescigno, R. Stein, R. Purple, and R. Poppele, "A neuronal model for the discharge patterns produced by cyclic inputs," *Bulletin of Mathematical Biophysics*, Vol. 32, 1970.

[30] F. Gantmacher, *The Theory of Matrices*, Chelsea, 1959.

[31] F. Juhász, "On the spectrum of a random graph," *Colloquia Mathematica Societatis János Bolyai*, Vol. 25, *Algebraic Methods in Graph Theory*, Szeged, Hungary, pp. 313-316, 1978.

[32] Z. Füredi and J. Komlós, "The eigenvalues of random symmetric matrices," *Combinatorica*, Vol 1, pp. 233-241, 1981.

CHAPTER 5

RECOGNITION OF NEURAL SIGNALS

5.1 INTRODUCTION

A first step in the study of the complex information processing mechanism of the brain is to examine how small populations of neurons interact. It is known that information is transferred from one neuron to another by means of a train of spike-shaped voltage pulses called action potentials, or simply spikes. Each neuron produces action potentials of fixed shape, and therefore the relevant information processing parameters are the time instants of the action potentials. Researchers are interested in devising techniques that allow to simultaneously record the spike instants of groups of adjacent neurons.

Monitoring the activity of a neuron is usually accomplished by placing an electrode outside the cell membrane. The electrode is a metal wire, insulated to its tip, or a fluid-filled tube. There are two types of extracellular recordings. The single-unit recording contains the activity of only one neuron, whereas the multi-unit recording consists of the activity of several neurons adjacent to the electrode. Multi-unit recordings are advantageous over single-unit recordings when trying to record the activity of a group of adjacent neurons. This is because the small scale of real neural networks makes inserting one recording electrode per cell impractical. Also, when using multi-electrode array chips, we do not have much control over the positions of the electrodes, and hence utilizing multi-unit recordings would be more appropriate in this case.

In spite of all the advantages of multi-unit recordings, they also present a rather serious waveform classification problem because the actual temporal sequence

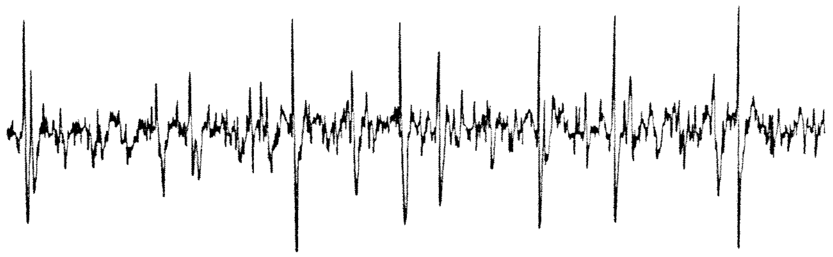

Fig. 5.1: An example of a multi-neuron recording

of action potentials in each individual neuron must be deciphered. A multi-unit recording contains the spike streams of several neurons adjacent to the electrode. Fortunately, spikes recorded from the same cell are more or less similar in shape, while spikes coming from different neurons usually have somewhat different shapes, depending on the neuron type, electrode characteristics, the distance between the electrode and the neuron, and the intervening medium. Fig. 5.1 illustrates some representative variations in spike shapes. It is our objective to detect and classify different spikes based on their shapes. We note that large spikes generally come from neurons adjacent to the electrode, while small spikes originate from remote neurons, as the spike waveform gets attenuated when traveling long distances.

The spike classification problem has been extensively studied in the last two decades (see [1]-[13], and see also the review paper by Schmidt [14]). There are two main categories of classification methods. The first one is the template matching technique. Templates representing the typical spike shapes of the different neurons are obtained. An unknown spike is compared to each of the templates, and the neuron corresponding to the closest template is chosen. In the second category, a small number of features are extracted from the spike waveform. Examples of

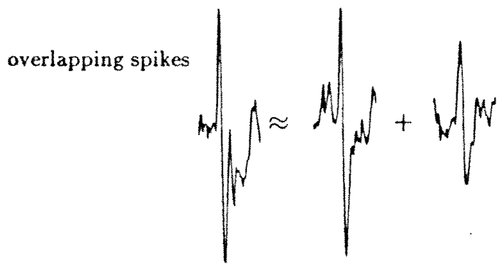

Fig. 5.2: An example of a temporal overlap of action potentials

features used are the positive peak value, the negative peak value, the time interval between the positive and negative peaks, and the peak-to-peak value.

One of the difficulties encountered in the classification problem is that action potentials from different neurons can overlap temporally producing novel waveforms (see Fig. 5.2 for an example of an overlap). To deal with these overlaps, we have first to detect the occurrence of an overlap, and then estimate the constituent spikes. Detecting overlapping spikes is potentially very important in understanding neural networks, as they can reveal correlations in the neural activity patterns, as well as cause and effect relationships. Unfortunately, only a few of the available spike separation algorithms consider these overlapping events. Those few tend to rely on heuristic rules and subtractive methods to resolve overlap cases. We propose a new off-line spike separation method, using an approach that minimizes the probability of error in classification, explicitly taking into account the likelihood of overlapping spikes (see also [15]).

The first step in classifying neural waveforms is obviously to identify the typical spike shapes occurring in a particular recording. To do this we have applied a learning algorithm on the beginning portion of the recording, which in an unsupervised fashion (i.e., without the intervention of a human operator) estimates the shapes. After the learning stage we have the classi ation stage, which is applied on the remaining portion of the recording. A new classification method is proposed, which gives minimum probability of error, even in case of the occurrence of overlapping spikes. Both the learning and the classification algorithms require a preprocessing step to detect the position of the spike candidate in the data record, as will be explained in the next section.

5.2 DETECTION

Spikes are usually characterized by an initial sharp rise. Therefore, most researchers use a simple level detecting algorithm [14], that signals a spike when recorded voltage levels cross a certain voltage threshold. However, variations in spike amplitude that are due to natural brain movements during recording (e.g., respiration) can cause the whole spike to be displaced downwards (or upwards), resulting in low (or high) positive and negative peaks. Thus, a level detector using a positive (or negative) threshold can miss some spikes. Alternatively, we have chosen to slide a window of fixed length until a time when the peak-to-peak value within the window exceeds a certain threshold.

5.3 LEARNING

Learning is performed on the beginning portion of the sampled data using the Isodata clustering algorithm [16]. The task is to estimate the number of neurons n whose spikes are represented in the waveform and to learn the different shapes of the spikes of the various neurons. For that purpose we apply the clustering algorithm choosing only one feature from the spike, the peak-to-peak value, which we have found to be quite an effective feature. Note that using the peak-to-peak value in the learning stage does not necessitate using it for classification (we might need additional or different features, especially for tackling the case of spike overlap). For completeness, the algorithm is explained in what follows:

As we do not know the number of available clusters (spike classes), we repeatedly perform the Isodata algorithm for 2 clusters, 3 clusters, and so on. Each time after performing the algorithm we measure

$$J(c) = \min_{i,j,i\neq j} |\mu_i - \mu_j|,$$

where c denotes the number of clusters, and μ_i is the mean of the spike peak-to-peak value of cluster i. If $J(c)$ is below a fixed threshold R, then we stop and take $c-1$ as the estimate of the number of available clusters (spike classes). The fixed threshold is taken as a constant times the maximum peak-to-peak value over all spikes in the learning stage (for example, we have chosen that constant as 0.1).

Let $y_1, ..., y_K$ represent the vectors of samples of the spikes in the learning period. Let $x_1, ..., x_K$ be the peak-to-peak values for these spikes. The algorithm is as follows:

1) $c = 2$ (number of clusters).

2) Generate $\mu_1, ..., \mu_c$ randomly (estimates of the means of the peak-to-peak values).

3) Classify $x_1, ..., x_K$ by assigning them to the nearest μ_i. Thus, we have c sets $A_1, ..., A_c$, where A_i represents the set of indices of the spikes assigned to cluster i (i.e., μ_i is the nearest mean).

4) Recompute the means

$$\mu_i = \sum_{j\epsilon A_i} x_j / K_i,$$

where K_i is the number of elements in A_i.

5) If any mean μ_i changed value in the previous step, go to 3).

6) Calculate $J = \min_{i,j,i \neq j} |\mu_i - \mu_j|$.

7) If $J < R$, then stop: the templates are $y^1, ..., y^{c-1}$.

8) Calculate typical template vectors of the different spike classes,

$$y^i = \sum_{j \epsilon A_i} y_j / K_i.$$

9) Increase c by 1. Go to 2).

5.4 CLASSIFICATION

Once we have identified the number of different spikes present, the classification stage is concerned with estimating the identities of the spikes in the recording, on the basis of the typical spike shapes obtained in the learning stage. In our classification scheme we make use of the information given by the shape of the detected spike as well as the firing rates (spike rates) of the different neurons. We note that LeFever and De Luca [17] have also developed a recognition technique for myoelectric signals, which exploits statistics about the firing rates. We propose here a different technique. Although the shape plays in general the most important role in the classification, the rates become a somewhat significant factor when dealing with overlapping events. This is because, in general, overlap is considerably less frequent than single spikes. The shape information is given by a set of features extracted from the waveform. Let $\mathbf{x}$ be the feature vector of the detected spike (e.g., the samples of the spike waveform). Let $N_1, ..., N_n$ represent the different neurons. The detection algorithm tells us only that at least one spike occurred in the narrow interval $(t - T_1, t + T_2)$ (= say I), where t is the instant of the peak of the detected spike, T_1 and T_2 are constants chosen subjectively according to the smallest possible time separation between two consecutive spikes, identifiable as two separate (non-overlapping) spikes. By definition then, if more than one spike occurs in the interval I, we have an overlap. As a matter of convention, the instant of the occurrence of a spike is taken to be that of the spike peak. For simplicity, we will consider the case of two possibly overlapping spikes, although the method can be extended easily to more. The classification rule that results in minimum probability of error is the one that chooses the neuron (or pair of neurons in case of overlap) having the highest probability of firing. We have therefore to compare the P_i's and the P_{lj}'s, defined as

$$P_i = P(N_i \text{ fired in } I|\mathbf{x}, A), \qquad i = 1, ..., n,$$

$$P_{lj} = P(N_l \text{ and } N_j \text{ fired in } I|\mathbf{x}, A), \qquad l, j = 1, ..., n, \quad j < l,$$

where A represents the event that one or two spikes occurred in the interval I. In other words, P_i is the probability that what has been detected is a single spike from neuron i, whereas P_{lj} is the probability that we have two overlapping spikes from neurons l and j (note that spikes from the same neuron never overlap). Henceforth we will use f to denote probability density. For the purpose of abbreviation, let $B_i(t)$ mean "neuron N_i fired at t," We will show that the classification problem can be reduced to comparing a number of likelihood functions.

Consider P_{lj}. We can write

$$P_{lj} = \int_{t-T_1}^{t+T_2} \int_{t-T_1}^{t+T_2} f(B_l(t_1), B_j(t_2)|\mathbf{x}, A) dt_1 dt_2.$$

We obtain, using Bayes rule,

$$P_{lj} = \int_{t-T_1}^{t+T_2} \int_{t-T_1}^{t+T_2} \frac{f(\mathbf{x}, A|B_l(t_1), B_j(t_2))}{f(\mathbf{x}, A)} f(B_l(t_1), B_j(t_2)) dt_1 dt_2.$$

Now, consider the two events $B_l(t_1)$ and $B_j(t_2)$. In the absence of any information about their dependence, we assume that they are independent. We get

$$f(B_l(t_1), B_j(t_2)) = f(B_l(t_1))f(B_j(t_2)).$$

Within the interval I, both $f(B_l(t_1))$ and $f(B_j(t_2))$ hardly vary because the duration of I is very small compared to a typical inter-spike interval. Therefore, we get the following approximation:

$$f(B_l(t_1)) \approx f(B_l(t))$$

$$f(B_j(t_2)) \approx f(B_j(t)).$$

The expression for P_{lj} becomes

$$P_{lj} \approx \frac{f(B_l(t))f(B_j(t))}{f(\mathbf{x}, A)} \int_{t-T_1}^{t+T_2} \int_{t-T_1}^{t-T_2} f(\mathbf{x}|B_l(t_1), B_j(t_2))dt_1 dt_2.$$

Notice that the term A was omitted from the argument of the density inside the integral, because the occurrence of two spikes at t_1 and $t_2 \epsilon I$ implies the occurrence of A. A similar derivation for P_i results in

$$P_i \approx \frac{f(B_i(t))}{f(\mathbf{x}, A)} \int_{t-T_1}^{t+T_2} f(\mathbf{x}|B_i(t_1))dt_1.$$

The term $f(\mathbf{x}, A)$ is common to all the P_{lj}'s and the P_i's. Hence, we can simply compare the following likelihood functions:

$$L_i = f(B_i(t)) \int_{t-T_1}^{t+T_2} f(\mathbf{x}|B_i(t_1))dt_1, \qquad i = 1, ..., n, \tag{5.1a}$$

$$L_{lj} = f(B_j(t))f(B_l(t)) \int_{t-T_1}^{t+T_2} \int_{t-T_1}^{t+T_2} f(\mathbf{x}|B_l(t_1), B_j(t_2))dt_1 dt_2,$$
$$l, j = 1, ..., n, \quad j < l. \tag{5.1b}$$

We propose two schemes for evaluating the terms $f(B_i(t))$. In the first one, we ignore the knowledge about the previous firing pattern except for the estimated firing rates $\lambda_1, ..., \lambda_n$ of the different neurons $N_1, ..., N_n$, respectively. Then the probability of a spike coming from neuron N_i in an interval of duration dt is simply $\lambda_i dt$. Hence,

$$f(B_i(t)) = \lambda_i. \tag{5.2}$$

In the second scheme we do not use any previous knowledge except for the total firing rate (of all neurons), say α. Then

$$f(B_i(t)) = \frac{\alpha}{n}. \tag{5.3}$$

Although the second scheme does not use as much of the information about the firing pattern as the first scheme does, it has the advantage of obtaining and using a more reliable estimate of the firing rate, because in general the overall firing rate changes less with time than the individual rates. Also, the estimate of α does not depend on previous classification results, but rather on the detection results, which are usually quite reliable. However, the second scheme is useful mostly when the firing rates of the different neurons do not differ much; otherwise the first scheme is preferred.

In real recording situations, sometimes we encounter voltage signals that are much different from any of the previously learned typical spike shapes or their pairwise overlaps. This can happen, for example, because of a falsely detected noise spike, a spike from a class not encountered in the learning stage, or to the overlap of three or more spikes. To cope with these cases, we use the reject option. This means that we refuse to classify the detected spike because of the unlikelihood of the assumed event A. The reject option is therefore employed whenever $P(A|\mathbf{x})$ is smaller than a certain threshold. We know that

$$P(A|\mathbf{x}) = f(A, \mathbf{x})/[f(A, \mathbf{x}) + f(A^c, \mathbf{x})],$$

where A^c is the complement of the event A. The density $f(A^c, \mathbf{x})$ can be approximated as uniform (over the possible values of $\mathbf{x}$) because a large variety of cases are covered by the event A^c. It follows that we can just compare $f(A, \mathbf{x})$ to a threshold. Hence, the decision strategy finaly becomes: Reject if the sum of the likelihood functions is less than a threshold. Otherwise, choose the neuron (or pair of neurons) corresponding to the largest likelihood functions. Note that the sum of the likelihood functions equals $f(A, \mathbf{x})$

Now, let us evaluate the integrals in (5.1). We take the samples of the spike (or possibly also just part of the samples) to constitute the feature vector $\mathbf{x}$. The added noise, partly thermal noise from the electrode and partly because of firings from distant neurons, can usually be approximated as white Gaussian. Let the variance be σ^2. It can be estimated from idle (non-spiking) periods of the recording. Overlapping spikes are assumed to add linearly. (Several researchers observed that this assumption holds well, e.g., [8].) The integrals in the likelihood functions can be approximated as summations (note, in fact, that we have samples available, not a continuous waveform). Let y^i represent the typical feature vector (template) associated with neuron N_i, with the m^{th} component being y^i_m. Then

$$f(\mathbf{x}|B_i(k+k_1)) = \frac{1}{(2\pi)^{M/2}\sigma^M} \exp\left[-\frac{1}{2\sigma^2}\sum_{m=1}^{M}(x_m - y^i_{m-k_1})^2\right]$$

$$f(\mathbf{x}|B_l(k+k_1), B_j(k+k_2)) = \frac{1}{(2\pi)^{M/2}\sigma^M} \exp\left[-\frac{1}{2\sigma^2}\sum_{m=1}^{M}(x_m - y^l_{m-k_1} - y^j_{m-k_2})^2\right],$$

where x_m is the m^{th} component of $\mathbf{x}$, and M is the dimension of $\mathbf{x}$.

To summarize, the recognition procedure becomes as follows:

1) Obtain the likelihood functions:

$$L'_i = f(B_i(k)) \sum_{k_1=-M_1}^{M_2} \exp\left[-\frac{1}{2\sigma^2}\sum_{m=1}^{M}(x_m - y^i_{m-k_1})^2\right], \tag{5.4a}$$

$$L'_{lj} = f(B_l(k))f(B_j(k)) \sum_{k_1=-M_1}^{M_2} \sum_{k_2=-M_1}^{M_2} \exp\left[-\frac{1}{2\sigma^2}\sum_{m=1}^{M}(x_m - y^l_{m-k_1} - y^j_{m-k_2})^2\right], \quad j < l, \qquad (5.4b)$$

where k is the spike instant, and the interval from $-M_1$ to M_2 corresponds to the interval I defined at the beginning of the section.

2) If $\sum_i L'_i + \sum_l \sum_{j<l} L'_{lj} < c$ (c is a constant), then reject (spike is unknown).

3) Otherwise, obtain $\max_{i,j,l}(L'_i, L'_{lj})$. The corresponding index (indices) represents (represent) the firing neuron(s).

We note that in the case of high signal-to-noise ratio the summations in (5.4a) and (5.4b) are dominated by the term corresponding to the closest distance between the vector of x_m's and the time-shifted templates. In such a situation we can simply use the following likelihood functions:

$$L''_i = 2\sigma^2 \ln\left[f(B_i(k))\right] - \min_{k_1}\left[\sum_{m=1}^{M}(x_m - y^i_{m-k_1})^2\right],$$

$$L''_{lj} = 2\sigma^2 \ln\left[f(B_l(k))\right] + 2\sigma^2 \ln\left[f(B_j(k))\right] - \min_{k_1,k_2}\left[\sum_{m=1}^{M}(x_m - y^l_{m-k_1} - y^j_{m-k_2})^2\right].$$

We observe similarities between this case and the template matching method. The presented method gives additional bias terms to incorporate the effects of firing rates. Note that the bias for the likelihood functions of overlapping spikes is strongly negative in case of low individual firing rates. This is consistent with the intuitive fact that in this case a detected event would be less likely an overlap, when the spike streams are independent.

We note that the discharge rates λ_i and α change with time. It is therefore imperative to adjust the rate estimates with time so as to track the changes. We use exponential averaging, as follows. In every block of H samples we count the number of spikes of class i (say it is D_i), and the overall number of spikes (say D). The new estimates are updated as follows:

$$\lambda_i(\text{new}) = a\lambda_i(\text{old}) + (1-a)D_i/H,$$

$$\alpha(\text{new}) = a\alpha(\text{old}) + (1-a)D/H,$$

where a is a constant between 0 and 1. The choice of a should depend on the choice of H. If H is low (H can be as small as 1), then a should be very close to 1, and vice versa.

5.5 IMPLEMENTATION

The techniques we have just described were tested in the following way. For the first experiment we identified two spike classes in a recording from the rat cerebellum. We created a signal composed of a number of spikes from the two classes at random instants, plus noise. To make the situation as realistic as possible, the added noise is taken from idle periods (i.e., non-spiking) of a real recording. The reason for using such an artificially generated signal was to be able to know the class identities of the spikes, in order to test our approach quantitatively. We implemented the detection and classification techniques on the obtained signal, with various values of noise amplitude. In our case the ratio of the peak-to-peak values of the templates turned out to be 1.375. Also, the spike rate of one of the classes was twice that of the other class. Fig. 5.3 shows the results with applying the first scheme (i.e., using Eq. 5.2). The overall percentage correct classification for all spikes (solid curve) and the percentage correct classification for overlapping spikes (dashed curve) are plotted versus the standard deviation of the noise σ normalized with respect to the peak h of the large template. Notice that the overall classification accuracy is near 100% for σ/h less than 0.15, which is actually the range of noise amplitudes we mostly encountered in our work with real recordings. Observe also the good results for classifying overlapping events. We have also applied the second scheme (i.e., using Eq. 5.3) and obtained more or less similar results. We wish to mention that the thresholds for detection and for the reject option are set up so as to obtain no more than 3% of falsely detected spikes.

A similar experiment is performed with three waveforms (three classes), where two of the waveforms are the same as those used in the first experiment. The third is the average of the first two. All three neurons have the same spike rate (i.e., $\lambda_1 = \lambda_2 = \lambda_3$). Hence, both classification schemes (Eq. (5.2) and Eq. (5.3)) are equivalent in this case. Fig. 5.4 shows the overall as well as the sub-category of overlap classification results. We observe that the results are worse than those for the two-class case. This is because the spacings between the templates are in general smaller. Notice also that the accuracy in resolving overlapping events is now tangibly less than the overall accuracy. However, we can say that the results are acceptable in the range of σ/h less than 0.1. From this experiment we observe also that for the case of high noise amplitude, the information given in the spike rates becomes considerably significant in preventing wrongly classifying single spikes as overlaps. This is because in our method the factor multiplying the summations in the L_{lj}'s is generally much less than that of L_i's, thus producing a frequently needed bias for single spikes.

Finally, in the last experiment the techniques are implemented on real recordings from the rat cerebellum. The recorded signal is band-pass-filtered in the frequency range 300 Hz - 10 KHz, then sampled with a rate of 20KHz. For classification, we take 20 samples per spike as features. Before performing the classification, the zero-level of the spike is set so that the positive peak and the negative peak are equal in magnitude. It has been noticed, and also verified, using experiments with single-unit electrodes, that sometimes the level of the spike gets displaced because of small movements during the recording. Fig. 5.5 shows the results of the proposed

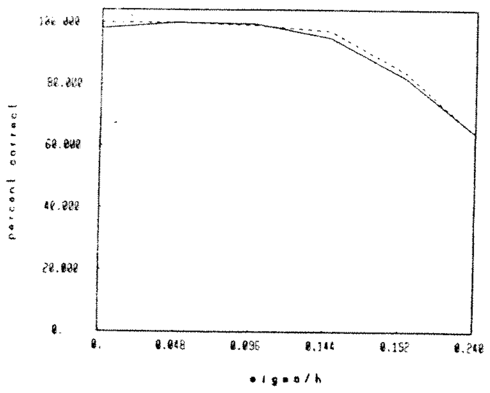

Fig. 5.3: Overall (solid curve) and overlap (dashed curve) classification accuracy for a two-class case.

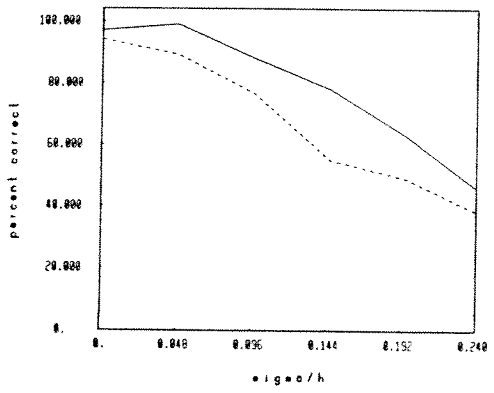

Fig. 5.4: Overall (solid curve) and overlap (dashed curve) classification accuracy for a three-class case.

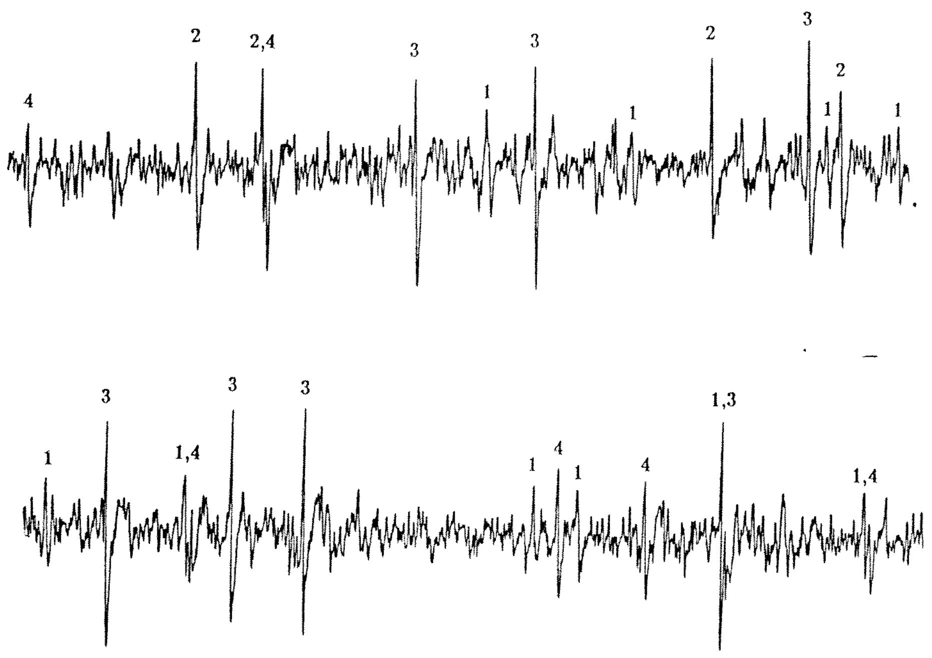

Fig. 5.5: Classification results for a recording from the rat cerebellum.

method, using the first scheme (Eq. 5.2). The number of neurons whose spikes are represented in the waveform is estimated to be four. The detection threshold is set up so that spikes that are too small are disregarded, because they come from several neurons far away from the electrode and are hard to distinguish. Notice the overlaps of classes 2 and 4, and 1 and 4 (twice), and 1 and 3, which were detected by the algorithm. Overall, the discrepancies between classifications done by the proposed method and an experienced human observer were found to be small.

5.6 CONCLUSION

Many researchers have considered the problem of spike classification in multi-neuron recordings, but only a few have tackled the case of spike overlap, which could occur frequently, particularly if the group of neurons under study is stimulated. In this work we propose a method for spike classification, which can also aid in detecting and classifying overlapping spikes. By taking into account the statistical properties of the discharges of the neurons sampled, this method minimizes the probability of classification error. The application of the method to artificial as well to as real recordings confirms its effectiveness.

REFERENCES

[1] G. Gerstein and W. Clarke,"Simultaneous studies of firing patterns in several neurons," *Science*, Vol. 143, pp. 1325-1327, 1964.

[2] M. Abeles and M. Goldstein, "Multispike train analysis," *Proc. IEEE*, Vol. 65, pp.762-773, 1977.

[3] D. Mishelevich, "On-line real-time digital computer separation of extracellular neuroelectric signals," *IEEE Trans. Bio-Med. Eng.*, BME-17, pp. 147-150, 1970.

[4] G. Dinning and A. Sanderson, "Real-time classification of multiunit neural signals using reduced feature sets," *IEEE Trans. Bio-Med. Eng.*, BME-28, pp. 804-812, 1981.

[5] V. Prochazka and H. Kornhuber, "On-line multi-unit sorting with resolutiion of superposition potentials," *Electroenceph. clin. Neurophysiol.*, Vol. 34, pp. 91-93, 1973.

[6] E. D'Hollander and G. Orban, "Spike recognition and on-line classification by unsupervised learning system," *IEEE Trans. Bio-Med. Eng.*, BME-26, pp. 279-284, 1979.

[7] W. Roberts, "Optimal recognition of neuronal waveforms," *Biol. Cybernet.*, Vol. 35, pp. 73-80, 1979.

[8] W. Roberts and D. Hartline, "Separation of multi-unit nerve impulse trains by a multi-channel linear filter algorithm," *Brain Res.*, Vol. 94, pp. 141-149, 1975.

[9] R. Stein, S. Andreassen, and M. Oguztoreli, "Mathematical analysis of optimal multi-channel filtering for nerve signals," *Biol. Cybern.*, Vol. 32, pp. 19-24, 1979.

[10] R. Millecchia and T. McIntyre, "Automatic nerve impulse identification and separation," *Comput. Biomed. Res.*, Vol. 11, pp. 459-468, 1978.

[11] J. Dill, P. Lockemann, and K. Naka, "An attempt to analyze multi-unit recordings," *Electroenceph. clin. Neurophysiol.*, Vol. 28, pp. 79-82, 1970.

[12] Y.-F. Wong, J. Banik, and J. Bower, "Neural networks for template matching: application to real-time classification of action potentials of real neurons," in *Neural Information Processing Systems*, D. Anderson, *Ed.*, American Institute of Physics, New York, pp. 103-113, 1987.

[13] B. Wheeler and W. Heetderks, "A comparison of techniques for classification of multiple neural signals," *IEEE Trans. Bio-Med. Eng.*, BME-29, pp. 752-759, 1982.

[14] E. Schmidt, "Computer separation of multi-unit neuroelectric data: a review,"

J. Neurosci. Methods, 12, pp. 95-111, 1984.

[15] A. Atiya and J. Bower, "Optimal neural spike classification," in *Neural Information Processing Systems*, D. Anderson, *Ed.*, American Institute of Physics, New York, pp. 95-102, 1987.

[16] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, John Wiley, 1973.

[17] R. LeFever and C. De Luca, "A procedure for decomposing the myoelectric signal into its constituent action potentials- part I: technique, theory, and implementation," *IEEE Trans. Bio-Med. Eng.*, BME-29, pp. 149-157, 1982.