Automatica, Vol. 28, No. 6, pp. 1083–1112, 1992
Printed in Great Britain.

0005-1098/92 $5.00 + 0.00
Pergamon Press Ltd
© 1992 International Federation of Automatic Control

Survey Paper

Neural Networks for Control Systems—A Survey*

K. J. HUNT,† D. SBARBARO,† R. ŻBIKOWSKI† and P. J. GAWTHROP†

Neural networks potentially provide a general framework for modelling and control of nonlinear systems. A survey of neural networks from a control systems perspective illustrates that deep insight into this potential can be achieved.

Key Words—Neural networks; nonlinear control systems; nonlinear systems modelling; systems identification.

Abstract—This paper focuses on the promise of artificial neural networks in the realm of modelling, identification and control of nonlinear systems. The basic ideas and techniques of artificial neural networks are presented in language and notation familiar to control engineers. Applications of a variety of neural network architectures in control are surveyed. We explore the links between the fields of control science and neural networks in a unified presentation and identify key areas for future research.

1. INTRODUCTION

No-one active in the field of control systems can be unaware of the growth of papers, journals, conferences and conference sessions devoted to the topic of Artificial Neural Networks. This is clearly indicative of a wide intellectual interest in the concepts associated with Artificial Neural Networks together with a desire to use the corresponding algorithms within a range of application areas. Control systems form one such application area as witnessed in part by recent special issues on the subject (*IEEE Control Systems*, 1988, 1989, 1990).

As with other growth areas in the past (for example, adaptive control and expert systems to name but two) it is possible to take two extreme positions either of which equally obstructs progress. The first such position is to embrace the new subject in an undiscerning way and use it unscientifically. This leads to papers of the form: "x is the hot topic of the decade and therefore supersedes all other approaches. We embedded it in a controller to give us an x controller which is, by definition, the best controller. We simulated it with impressive results". Here, x is self-tuning, expert system, Artificial Neural Network or any other hot topic. Such papers tend to advance anthropomorphic arguments to explain apparent successes. The second extreme is to dismiss such growth areas as consisting of mere hyperbole and not attempt to use them.

Both these extreme positions must be avoided if progress is to be made. Instead, a rational evaluation of new ideas in the context of established techniques is needed. This should avoid the two extremes and rather steer a middle course allowing new ideas and established principles to fuse thus allowing real (as opposed to illusory) advances to be made.

Those who have read Wiener's seminal book "Cybernetics" (Wiener (1948)) will know that control, information and neural science were once regarded as a common subject under the banner of Cybernetics. Since that time, the disciplines of control, computing science (including Artificial Intelligence) and neurobiology have tended to go their own separate ways. This has lead to an unfortunate breakdown in communication between the disciplines; in particular the differing jargon and notation now poses a barrier to effective interchange of ideas.

This survey is a contribution to removing this barrier for researchers currently working in the area of control systems in that we attempt to present the basic ideas and techniques of Artificial Neural Networks in language and notation familiar to control engineers. This has the twofold advantage of making such techniques more readily accessible to control engineers and, conversely, allowing control systems insights to be applied to Artificial Neural Network techniques.

We note that an early comprehensive survey of adaptive and learning systems and their application in control and pattern analysis can be found in the classic book by Tsypkin (1971).

1.1. *Origins of connectionist research*

In recent years there has been an increasing interest in studying the mechanisms and structure of the brain. This has led to the development of new computational models, based on this biological background, for

* Received 25 June 1991; revised 24 February 1992; revised 2 April 1992; received in final form 14 May 1992. The original version of this paper was not presented at any IFAC meeting. This paper was recommended for publication in revised form by Editor K. J. Åström.

† Control Group, Department of Mechanical Engineering, University of Glasgow, Glasgow G12 8QQ, U.K.

solving complex problems like pattern recognition, fast information processing and adaptation.

In the early 1940s the pioneers of the field (McCulloch and Pitts (1943)) studied the potential and capabilities of the interconnection of several basic components based on the model of a neuron. Others, like Hebb (1949), were concerned with the adaptation laws involved in neural systems. Rosenblatt (1958) coined the name Perceptron and devised an architecture which has subsequently received much attention. Minsky and Papert (1969) introduced a rigorous analysis of the Perceptron; they proved many properties and pointed out limitations of several models. In the 1970s the work of Grossberg came to prominence (Grossberg (1976)). This work, based on biological and psychological evidence, proposed several architectures of nonlinear dynamic systems with novel characteristics. Hopfield applied a particular nonlinear dynamic structure to solve technical problems like optimization (Hopfield (1982)). In 1986 the parallel distributed processing (PDP) group published a series of results and algorithms (Rumelhart *et al.* (1986)). This work gave a strong impulse to the area and provided the catalyst for much of the subsequent research in this field. For a fine collection of key papers in the development of models of neural networks see *Neurocomputing: Foundations of Research* (Anderson and Rosenfeld (1988)). Since the work of the PDP group much research has been done and today there are several well-defined architectures to tackle a variety of problems. Many examples of real-world applications ranging from finance to aerospace are currently being explored (Hecht-Nielsen (1988)).

1.2. *Neural networks and control*

To provide a rational assessment of new methods it is important to compare and contrast the emerging technologies with established and traditional techniques. One way to do this is to list the key characteristics of competing methods in the context of the field under study.

With specific reference to neural networks in control the following characteristics and properties of neural networks are important:

- Nonlinear systems. Neural networks have greatest promise in the realm of nonlinear control problems. This stems from their theoretical ability to approximate arbitrary nonlinear mappings. Networks may also achieve more parsimonious modelling than alternative approximation schemes; this question requires further study.
- Parallel distributed processing. Neural networks have a highly parallel structure which lends itself immediately to parallel implementation. Such an implementation can be expected to achieve a higher degree of fault tolerance than conventional schemes. The basic processing element in a neural network has a very simple structure. This, in conjunction with parallel implementation, results in very fast overall processing.
- Hardware implementation. This is closely related to the preceding point. Not only can networks be implemented in parallel, a number of vendors have recently introduced dedicated VLSI hardware implementations. This brings additional speed and increases the scale of networks which can be implemented.
- Learning and adaptation. Networks are trained using past data records from the system under study.•A suitably trained network then has the ability to generalize when presented with inputs not appearing in the training data. Networks can also be adapted on-line.
- Data fusion. Neural networks can operate simultaneously on both quantitative and qualitative data. In this respect networks stand somewhere in the middle ground between traditional engineering systems (quantitative data) and processing techniques from the artificial intelligence field (symbolic data).
- Multivariable systems. Neural networks naturally process many inputs and have many outputs; they are readily applicable to multivariable systems.

It is clear that a modelling paradigm which has all of the above features has great promise.

From the control theory viewpoint the ability of neural networks to deal with nonlinear systems is perhaps most significant. The great diversity of nonlinear systems is the primary reason why no systematic and generally applicable theory for nonlinear control design has yet evolved. A range of "traditional" methods for the analysis and synthesis of nonlinear controllers for specific classes of nonlinear systems exist: phase plane methods, linearization techniques and describing functions are three examples.

However, it is the ability of neural networks to represent nonlinear mappings, and hence to model nonlinear systems, which is the feature to be most readily exploited in the synthesis of nonlinear controllers.

In this respect more recent methods of nonlinear control design including nonlinear operator–theoretic methods and optimization techniques have an important role to play. One promising possibility is to use neural networks to provide the nonlinear system models required by these techniques. This possibility is explored in depth in Section 5.3.

A further established area in control science with great relevance here is adaptive systems theory. This area has produced many fruitful theoretical results in the past 15 years. Although the established adaptive systems theory is founded upon the assumption of linear time-invariant systems, the concepts and theoretical challenges addressed are likely to find strong parallels in the neural networks field. As emphasised in Section 7, however, the nonlinear nature of the subsystems involved in the neural networks case means that the problems encountered are likely to be more complex and many fundamental theoretical questions will need to be addressed.

The compilation book (Miller *et al.* (1990)) provides a broad overview of the field of neural networks in control. That book also contains a number of benchmark problems.

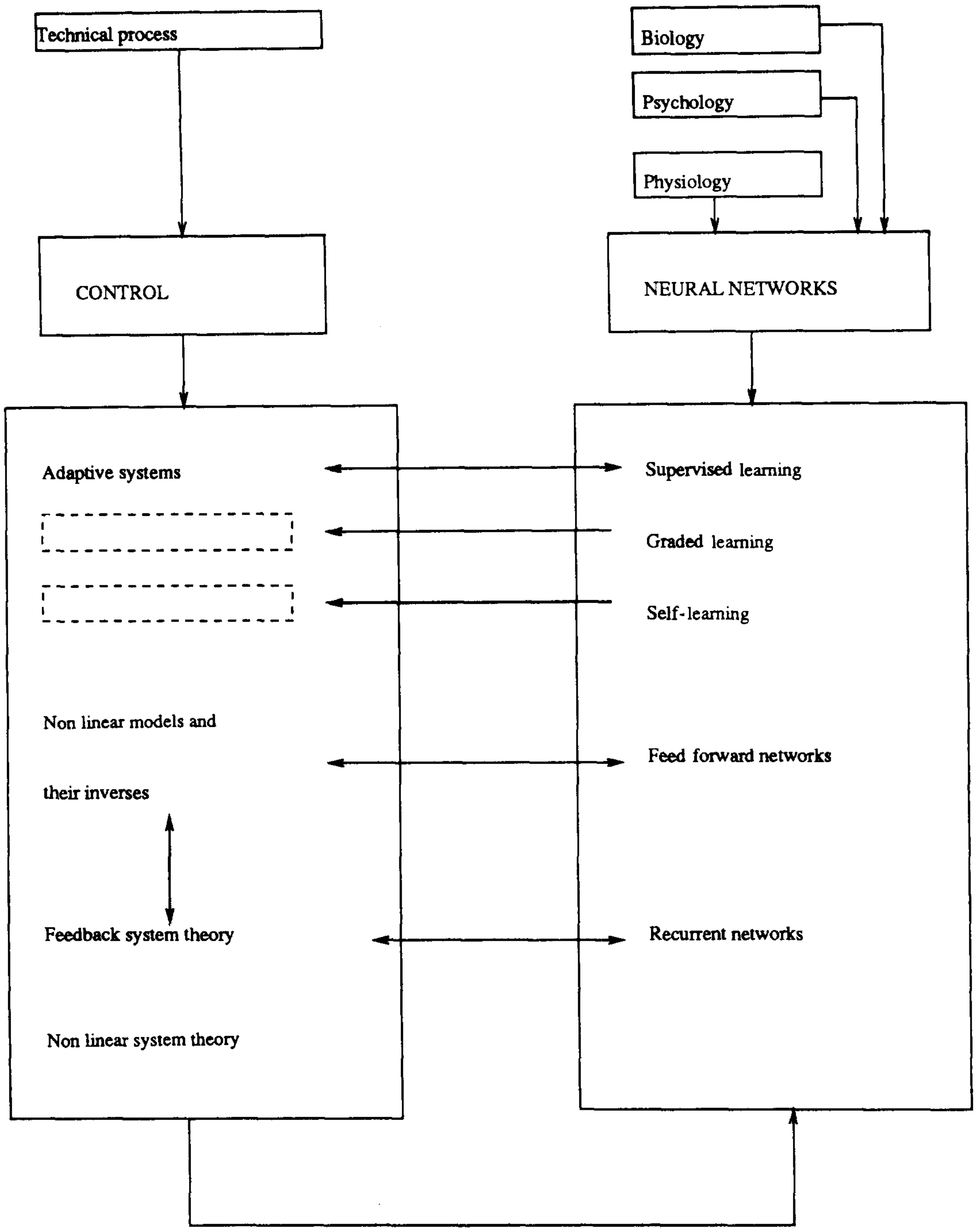

FIG. 1. Relation between control and neural networks.

Our view of the general relationship between the fields of control science and neural networks is shown in Fig. 1. Blank boxes are used when no obvious parallel exists. The technical terms used in the diagram are defined and discussed in later sections of the paper. In this paper we attempt to expand upon the links between the two fields in a unified presentation, and to identify key areas for future research. Key abbreviations used in the paper are summarised in Table 1.

TABLE 1. ABBREVIATIONS USED IN THE PAPER

Abbreviation	Meaning
ANN	Artificial neural network
ART	Adaptive resonance theory
BPTT	Backpropagation through time
CAM	Content-addressable memory
CMAC	Cerebellar model articulation controller
CMN	Cellular neural network
CPN	Counterpropagation network
IMC	Internal model control
LVQ	Learning vector quantization

1.3. *Organization of the paper*

The paper has two main parts:

- An exposition of Artificial Neural Network (ANN) techniques in the notation and language of control systems. Section 2 discusses the architecture of ANNs. Section 3 (static networks) and Section 4 (dynamic networks) discuss how these can be made adaptive and trained using learning algorithms.

- A comparative survey of Artificial Neural Network applications in control systems set in the context of the first part. Section 5 discusses system representation and modelling by ANNs and applies these representations to system identification and hence to nonlinear and adaptive control. Section 6 discusses a variety of further applications of ANNs.

In Section 7 we discuss a range of future theoretical research directions for ANNs in the context of control.

2. NETWORK ARCHITECTURES

The network architecture is defined by the basic processing elements and the way in which they are interconnected. These two aspects are discussed in turn.

2.1. *Basic elements*

The basic processing element of the connectionist architecture is often called a neuron by (loose) analogy with neurophysiology, but other names such as perceptron (Rosenblatt (1958)) or adaline (Widrow and Hoff (1960)) are also used. Below we describe a standard and unifying model for neural networks and later in this section show that many well-known structures fall within the standard model. Many of the basic processing elements may be considered to have three components (see Fig. 2):

(1) A weighted summer.
(2) A linear dynamic SISO system.
(3) A non-dynamic nonlinear function.

These elements are considered in turn in the following sections.

2.1.1. *Weighted summer.* The *weighted summer* is described by

$$v_i(t) = \sum_{j=1}^{N} a_{ij} y_j(t) + \sum_{k=1}^{M} b_{ik} u_k(t) + w_i, \qquad (1)$$

giving a weighted sum v_i in terms of the outputs of all elements y_j, external inputs u_k and corresponding weights a_{ij} and b_{ik} together with constants w_i. N of these weighted summer elements can be conveniently expressed in vector-matrix notation.

Stacking N weighted sums v_i into a column vector v, the N outputs y_j into a vector y and M inputs u_k into a vector u and the N constants w_i into a vector w, equation (1) may be rewritten in vector matrix form as:

$$v(t) = Ay(t) + Bu(t) + w, \qquad (2)$$

where the ijth element of the $N \times N$ matrix A is a_{ij} and the ikth element of the $N \times M$ matrix B is b_{ik} (the constants w_i could be incorporated with the inputs u_k, but it is useful to represent them explicitly).

2.1.2. *Linear dynamic system.* The linear dynamic SISO system has input v_i and output x_i. In transfer function form it is described by

$$\bar{x}_i(s) = H(s)\bar{v}_i(s), \qquad (3)$$

where the bar denotes Laplace transformation. In the time domain, equation (3) becomes:

$$x_i(t) = \int_{-\infty}^{t} h(t-t')v_i(t')\,\mathrm{d}t', \qquad (4)$$

where $H(s)$ and $h(t)$ form a Laplace transform pair.

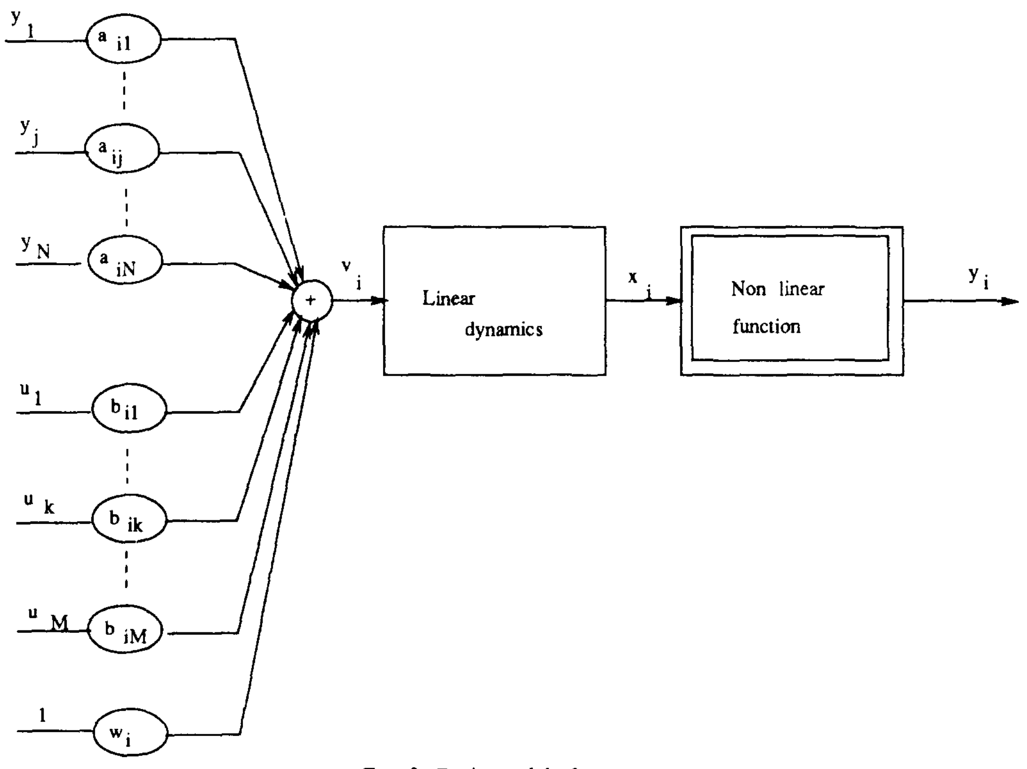

FIG. 2. Basic model of a neuron.

Five common choices of $H(s)$ are:

$$
\begin{aligned}
H(s) &= 1, \\
H(s) &= \frac{1}{s}, \\
H(s) &= \frac{1}{1+sT}, \\
H(s) &= \frac{1}{\alpha_0 s + \alpha_1}, \\
H(s) &= e^{-sT},
\end{aligned}
\tag{5}
$$

corresponding to

$$
\begin{aligned}
h(t) &= \delta(t), \\
h(t) &= \begin{cases} 0 & t<0 \\ 1 & t \geq 0 \end{cases}, \\
h(t) &= \frac{1}{T} e^{-t/T}, \\
h(t) &= \frac{1}{\alpha_0} e^{-(\alpha_1/\alpha_0)t}, \\
h(t) &= \delta(t-T),
\end{aligned}
\tag{6}
$$

where δ is the Dirac delta function. In the time domain the corresponding input–output relations are:

$$
\begin{aligned}
x_i(t) &= v_i(t), \\
\dot{x}_i(t) &= v_i(t), \\
T\dot{x}_i(t) + x_i(t) &= v_i(t), \\
\alpha_0 \dot{x}_i(t) + \alpha_1 x_i(t) &= v_i(t), \\
x_i(t) &= v_i(t-T).
\end{aligned}
\tag{7}
$$

The first, second, and third versions are clearly just special cases of the fourth.

Discrete-time dynamic systems are also used. For example

$$\alpha_0 x_i(t+1) + \alpha_1 x_i(t) = v_i(t), \tag{8}$$

where t is now an integer time index.

2.1.3. *Non-dynamic nonlinear function.* The non-dynamic nonlinear function $g(\cdot)$ gives the element output y_i in terms of the transfer function output x_i:

$$y_i = g(x_i). \tag{9}$$

There are a number of twofold classifications of these functions:

(1) Differentiable/non-differentiable.

(2) Pulse-like/step-like.

(3) Positive/zero-mean.

Classification 1 distinguishes smooth from sharp functions. Smooth functions are needed for some adaptation algorithms such as backpropagation (Rumelhart *et al.* (1986)) (Section 3.2), whereas discontinuous (e.g. signum) functions are needed to give a true binary output.

Classification 2 distinguishes functions which only have a significant output value for inputs near to zero from functions which only change significantly around zero.

Classification 3 refers to step like functions. Positive functions change from 0 at $-\infty$ to 1 at ∞; zero-mean changes from -1 at $-\infty$ to 1 at ∞.

Some standard functions are given in Table 2. Note that in the table there are strong relations between the given functions. The sigmoid and tanh functions are similar: sigmoid ranges from 0 to 1 while tanh ranges from -1 to 1. Secondly, the threshold functions correspond to the high gain limits of the sigmoid and tanh functions.

It is of course possible to create pulse-like functions from differentiable step-like functions by differentiation, and vice-versa, although this does not often seem to be done. Alternatively, pulse-like functions may be created by differentiating shifted step-like functions. This is sometimes achieved by interconnecting two neurons appropriately.

It is known that neurons located in different parts in the nervous system have different characteristics. For example, the neurons of the ocular motor system have a sigmoid characteristic while those located in the visual area have a Gaussian characteristic (Ballard (1988)). The former is called variable-encoding and exhibits monotonic behaviour. The second exhibits a response maximum that spans the measurement space. This is called value-encoded (Ballard (1988)). There are other types of function such as logarithmic and exponential which are useful (Daunicht (1990)), although their biological basis has not been established.

Each representation of the basic unit has advantages and some disadvantages. For example, the neuron with Gaussian encoding seems to have the ability to represent multidimensional variables and functions more easily than the sigmoid. This is further discussed in Section 5.

2.2. *Connections*

The neurons by themselves are not very powerful in terms of computation or representation but their interconnection allows us to encode relations between the variables giving different powerful processing capabilities. The three components of the neuron discussed in Section 2.1 can be combined in various ways. For example, if the neurons are all non-dynamic

TABLE 2. NONLINEAR FUNCTIONS $g(x)$

Name	Formula	Characteristics
Threshold	$+1$ if $x>0$ else 0	Non-differentiable, step-like, positive
Threshold	$+1$ if $x>0$ else -1	Non-differentiable, step-like, zero-mean
Sigmoid	$1/1+e^{-x}$	Differential, step-like, positive
Hyperbolic tangent	$\tanh(x)$	Differential, step-like, zero-mean
Gaussian	$e^{(-x^2/\sigma^2)}$	Differential, pulse-like

($H(s) = 1$) then an assembly of neurons can be written as the set of *algebraic* equations obtained by combining equations (2), (3), and (9):

$$x(t) = Ay(t) + Bu(t) + w, \quad y(t) = g(x(t)), \tag{10}$$

where x is a vector of Nx_i elements and $g(x)$ is a vector whose components are $g(x_i)$. If, on the other hand, each neuron has first order low-pass dynamics $H(s) = \frac{1}{Ts+1}$ then an assembly of neurons can be written as the set of *differential* equations:

$$T\dot{x}(t) + x(t) = Ay(t) + Bu(t) + w, \quad y(t) = g(x(t)). \tag{11}$$

Clearly, the solutions of equation (10) form possible steady-state solutions of equation (11).

A discrete-time version of equation (11) is

$$Tx(t+1) + (1-T)x(t) = Ay(t) + Bu(t) + w, \quad y(t) = g(x(t)), \tag{12}$$

where t is the integer time index.

The behaviour of such a network clearly depends on the interconnection matrix A and on the form of $H(s)$. Some important forms of the interconnection matrix are discussed in the following sections.

2.3. *Static multi-layer feedforward networks*

The connection of several layers gives the possibility of more complex nonlinear mapping between the inputs and the outputs.

This capability can be used to implement classifiers or to represent complex nonlinear relations among the variables (Section 5.1).

Such networks are typically non-dynamic; that is $H(s) = 1$ in equation (3). The connection matrix A is such that the outputs are partitioned into layers so that a neuron in one layer receives inputs only from neurons in the previous layer (or, in the case of the first layer, from the network input). There is no feedback in such networks. For example, in a three layer network, each layer containing N neurons, we may partition the network x, y, u and w vectors from equation (10) as:

$$\begin{bmatrix} x^1(t) \\ x^2(t) \\ x^3(t) \end{bmatrix} = A \begin{bmatrix} y^1(t) \\ y^2(t) \\ y^3(t) \end{bmatrix} + B \begin{bmatrix} u^1(t) \\ u^2(t) \\ u^3(t) \end{bmatrix} + \begin{bmatrix} w^1 \\ w^2 \\ w^3 \end{bmatrix}, \tag{13}$$

where the superscripts denote the corresponding layer in the network. The structure of the A and B matrices for this network are as follows:

$$A = \begin{bmatrix} 0_{NN} & 0_{NN} & 0_{NN} \\ A^2 & 0_{NN} & 0_{NN} \\ 0_{NN} & A^3 & 0_{NN} \end{bmatrix}; \quad B = \begin{bmatrix} B^1 & 0_{NM} & 0_{NM} \\ 0_{NM} & 0_{NM} & 0_{NM} \\ 0_{NM} & 0_{NM} & 0_{NM} \end{bmatrix}, \tag{14}$$

where 0_{NN} is the $N \times N$ zero matrix and 0_{NM} the $N \times M$ zero matrix. A^2 and A^3 are $N \times N$ matrices of weights while B^1 is a $N \times M$ matrix of weights. For the first layer we have

$$x^1(t) = B^1u^1(t) + w^1, \quad y^1(t) = g(x^1(t)), \tag{15}$$

and for the second and third layers

$$x^l(t) = A^l y^{l-1}(t) + w^l, \quad y^l(t) = g(x^l(t)), \tag{16}$$

where $l = 2, 3$.

Different characteristics are obtained using different nonlinearities $g(\cdot)$ from Table 2. Use of the sigmoid function is traditional Rumelhart *et al.* (1986).

2.4. *Dynamical networks*

The introduction of feedback produces a dynamic network with several stable points. The general equation can be expressed as

$$\dot{x}(t) = F(x(t), u(t), \theta), \quad y(t) = G(x(t), \theta). \tag{17}$$

Here, x represents the state, u the external inputs, and θ the parameters of the network. F is a function that represents the structure of the network and G is function which represents the relation between the state variables and the outputs.

Originally, feedback (recurrent) networks were introduced in the context of associative or content-addressable memory (CAM) problems (Kohonen (1987); Hopfield (1984)) for pattern recognition. The uncorrupted pattern is used as a stable equilibrium point and its noisy versions should lie in its basin of attraction. In this way, a dynamical system associated with a set of patterns is created. If the whole working space is corrrectly partitioned by such a CAM, then any initial condition (corresponding to a sample pattern) should have a steady-state solution corresponding to the uncorrupted pattern. The dynamics of such a classifier serve as a filter.

2.4.1. *Hopfield nets.* The best-known example of a CAM is the Hopfield net. Hopfield published two fundamental papers (Hopfield (1982, 1984)). However, the discrete model in Hopfield (1982) is different from the continuous one elaborated in Hopfield (1984).

The discrete model assumes the step-like nonlinearity

$$g(x_i(t)) \rightarrow \begin{cases} 1, & \text{if } x_i(t) > 0; \\ 0, & \text{if } x_i(t) < 0 \end{cases} \quad \text{with}$$

$$x_i(t) = \sum_{j=1}^{N} a_{ij} y_j(t) - w_i, \tag{18}$$

and works in the asynchronous mode, i.e. only one neuron output is calculated at a time, leaving the others unchanged. The active neuron, p, is chosen randomly. The systems evolves with weights a_{ij} established earlier (with the Hebbian rule, see below) and held fixed during output calculation. The update rule is as follows:

$$y_i(t+1) = \begin{cases} g(x_i(t)), & \text{if } i = p, x_i(t) \neq 0, \\ y_i(t), & \text{otherwise.} \end{cases} \tag{19}$$

If $a_{ii}=0$ and $a_{ij}=a_{ji}$ then the function

$$E(y)=-\tfrac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}a_{ij}y_iy_j+\sum_{i=1}^{N}w_iy_i \quad \text{or} \quad E(y)=-\tfrac{1}{2}y^TAy+w^Ty, \tag{20}$$

will decrease with every asynchronous change of y_p according to

$$\Delta E=-\Delta y_p\left[\sum_{j=1}^{N}a_{pj}y_j-w_p\right], \tag{21}$$

where $\Delta y_p(t)=y_p(t+1)-y_p(t)$. The expression (20) is *not* a Lyapunov function, since it is a quadratic form with an indefinite matrix (the zero diagonal). Moreover, Lyapunov theory for difference equations cannot be applied here, because the system of equations (18) is solved one equation at a time. However, the network will always reach an equilibrium because (20) is bounded and (21) is non-positive, and the system does not change when $\Delta E=0$. It will settle after a finite time, since the domain of E is finite.

The Hebbian rule is an attempt to encode P patterns, y^k, $k=1,\ldots,P$, as equilibrium points of the system represented by equations (18)–(19) by choosing

$$a_{ij}=\begin{cases}\sum_{k=1}^{P}(2y_i^k-1)(2y_j^k-1), & \text{if } i\neq j,\\ 0, & \text{otherwise,}\end{cases}$$

$$\text{and} \quad w_i=\tfrac{1}{2}\sum_{j=1}^{N}a_{ij}, \tag{22}$$

since the E is, within a multiple and a constant,

$$E\sim-(2y-\vec{1})^T\left[\sum_{k=1}^{P}(2y^k-\vec{1})^T(2y^k-\vec{1})\right](2y^k-\vec{1}), \tag{23}$$

where $\vec{1}$ is a vector of 1s. If the $(2y^k-\vec{1})$ are all orthogonal, E will have a minimum at each y^k and, hopefully, the dynamics of the system will have a region of attraction about each y^k that associates initial values of y that are near y^k and y^k. Detailed discussion of these issues can be found in Bruck (1989). It should be remarked here that the Hebbian rule encoding the patterns generates automatically the equilibrium points of (18)–(19), placing them in the locations resulting from y^k and (22) and unknown *a priori* to the designer.

The name Hopfield net is widely used for the continuous model described by

$$\begin{aligned}T_i\dot{x}_i&=-x_i+\sum_{j=1}^{N}a_{ij}y_j+u_i,\\ y_i&=g_i(x_i),\quad i=1,\ldots,N,\end{aligned} \tag{24}$$

where $x_i=x_i(t)$, $y_i=y_i(t)$ and $g_i(\cdot)$ are sigmoid functions. The system (24) is, unlike (18), just a system of ordinary differential equations. Hopfield suggested that Lyapunov function

$$E=-\tfrac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}a_{ij}y_iy_j+\sum_{i=1}^{N}\rho_i\int_0^{y_i}g_i^{-1}(\xi)\,d\xi-\sum_{i=1}^{N}u_iy_i, \tag{25}$$

where $\rho_i>0$ are constants, $g_i(\cdot)$ are monotone increasing functions and $a_{ij}=a_{ji}\ \forall i,j$. It is straightforward to show that $\dot{E}\leq 0$.

According to Hopfield a_{ij} can be of both signs (or 0), which means A can be indefinite in the general case. Neglected discussion of positivity conditions for (25) in Hopfield (1984) was addressed in Michel *et al.* (1989) and Li *et al.* (1988) where alternative means of analysis were described.

2.4.2. *Cohen–Grossberg Theorem.* The central issue of dynamic networks stability did not receive as much attention as did simulations and experiments. Hopfield nets stability, even after the work of Michel *et al.*, Li *et al.* (1988) and Michel *et al.* (1989) does not seem to be perfectly clear. The most general result in the area, the Cohen–Grossberg Theorem (Cohen and Grossberg (1983)), was obtained under conservative assumptions (see below), often violated by practitioners without much harm (Pineda (1987)). Other special cases of stability are also considered in the literature Sudharsanan and Sundareshan (1990), Kelly (1990), Tan *et al.* (1990) and Johnson (1991) and Poteryaiko (1991) to name a few. Below we present the Cohen–Grossberg Theorem and relate it to Hopfield nets stability.

The Cohen–Grossberg result applies to the general model

$$\dot{x}_i=r_i(x_i)\left[s_i(x_i)-\sum_{j=1}^{N}a_{ij}g_j(x_j)\right],\quad i=1,\ldots,N. \tag{26}$$

The following assumptions are made for $i,j=1,\ldots,N$:

1° $a_{ij}\geq 0$ and $a_{ij}=a_{ji}$;

2° $r_i(\xi)$ is continuous for $\xi\geq 0$ and positive for $\xi>0$;

3° $s_i(\xi)$ is continuous;

4° $g_i(\xi)\geq 0$ for $\xi\in(-\infty,+\infty)$, differentiable and monotone non-decreasing for $\xi\geq 0$;

5° $\limsup_{\xi\to\infty}[s_i(\xi)-a_{ii}g(\xi)]<0$;

6° either

a) $\lim_{\xi\to 0^+}s_i(\xi)=\infty$,

or

b) $\lim_{\xi\to 0^+}s_i(\xi)<\infty$ and $\int_0^{\varepsilon}\frac{d\xi}{r_i(\xi)}=\infty$ for some $\varepsilon>0$.

The assumption of weights symmetry, 1°, is popular in dynamic network models, but here it is strengthened by the non-negativity of a_{ij}. The continuity assumptions 2°–3° and positivity $r_i(\cdot)$ for positive argument are needed together to prove that if the system (26) starts from positive initial conditions then its trajectories remain positive. Thus the state space is contracted to the positive orthant $\mathcal{R}_+^N$, which together with the constraint $a_{ij}\geq 0$ is a strong condition. The assumptions about network nonlinearity, 4°, are mild in practice, unless the step-like function is used.

The Lyapunov function (recall 1°, 4° and $\mathcal{R}_+^N$

restriction) for the system (26) is defined as

$$V(x) = -\sum_{i=1}^{N} \int_0^{x_i} s_i(\xi) g_i'(\xi)\,\mathrm{d}\xi + \tfrac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij} g_i(x_i) g_j(x_j), \quad (27)$$

and its time derivative has the property $\dot{V}(x) \leq 0$, since it has the form

$$\dot{V}(x) = -\sum_{i=1}^{N} r_i(x_i) g_i'(x_i) \left[s_i(x_i) - \sum_{j=1}^{N} a_{ij} g_j(x_j) \right]^2. \quad (28)$$

In (28) Assumptions 2^0 and 4^0 were applied.

The price for the generality of the theorem is the introduction of technical (5° and 6°) and strong (1°) conditions, often violated in practice. The violations do not always result in instability, because the theorem is a sufficient condition.

2.5. *Other architectures*

There exist numerous neural network architectures and learning algorithms. Many of them are refinement/extensions of the ones described in this paper. In this section we include some of the original schemes with certain potential for control/optimization applications.

2.5.1. *Simulated annealing and Boltzmann machines.* Artificial neural networks learning is an optimization process (see Section 4.3.3), aimed at minimization of the error function E with respect to weights a_{ij}. Due to the nonlinearity of networks, $E(A)$ possesses many local minima (which is desirable for CAM, but not for control), resulting in suboptimal solutions when gradient methods are applied. An alternative approach is to allow occasional search directions increasing E, in this way trying to escape local minima.

The method is possible in the stochastic context (Metropolis *et al.* (1953)), inspiration coming from mathematical models describing melted metals annealing (Aarts and Korst (1989)). Originally, the physical interpretation says that in a high temperature (T) the probability that metal is in a given energy state is uniform. With the temperature dropping, the average chaotic movement of atoms becomes less intense, so that minima surrounded by "high" energy barriers are more likely for metal to assume. Provided the cooling process is slow, initial big perturbations (with T high) should allow the system to escape "shallow" local minima and, after lowering the temperature T, small perturbations should prevent it from jumping out of a "deep" (possibly global) minimum. Thus, controlling one parameter, T, it is possible to change "energy" probability distribution to steer the system toward the global solution, exciting it randomly to overcome local ones.

The Boltzmann machine (Ackley *et al.* (1985)) is an application of these ideas to the asynchronous Hopfield net (see Section 2.4.1). Interpreting E in (20) as an "energy" function, the key element is the calculation of ΔE of (21). The algorithm is as follows. Excite the network (18) (weights fixed) with a random initial condition and set the parameter T ("temperature") to a high value. Pick a random node, p. Calculate ΔE from (21) by setting $\Delta y_p = 1$. Generate a random number, $\gamma \in [0, 1]$ (uniform distribution). If $\gamma \leq (1 + \exp(\Delta E/T))^{-1}$ set $y_p(t+1) = 1$. After a prescribed number of changes of E and/or x, T is lowered and the process is started over from the state it reached in the previous iteration. In the equilibrium the relative probability of two global states y^{k_1}, y^{k_2}, will follow the Boltzmann distribution $P(y = y^{k_1})/P(y = y^{k_2}) = \exp([E(y^{k_2}) - E(y^{k_1})]/T)$.

Since E is non-increasing (see Section 2.4.1) the network will head to the nearest local minimum. However, if T is high then $\gamma \geq \exp(-\Delta E/T)$ will hold more often than not. The system will be perturbed on the way to a local solution and pushed randomly (with the probability $\exp(-\Delta E/T)$) towards another local minimum. This "dither signal" will fade away with dropping temperature T, possibly allowing the network to settle close to the global extremum.

The method is statistically convergent but this requires an infinite number of trials and continuous change of T. In practice this is violated, so *ad hoc* methods of termination must be employed (e.g. $|\Delta E| < \epsilon$). This, however, prevents the network from reaching the true global solution. Because of its trial-and-error character the method is very slow.

2.5.2. *Lateral connections.* Lateral connections are connections among units in the same layer, as shown in Fig. 3. This type of connection is vital in many different applications, such as maximum selection, competitive learning and contrast enhancement. There are two kinds of lateral connections: non-recurrent and recurrent. The first class can be described using

$$A = 0,$$
$$B = B_1 - B_2,$$

where $A = 0$ implies no feedback, B_1 is the identity matrix, and B_2 (having diagonal entries equal to zero) represents explicitly the inhibition of other inputs. This kind of lateral connection has been shown to be useful in estimating acoustic spectra (Shamma (1989)). The recurrent architecture uses the interconnection matrix A to provide inhibition. In this case we

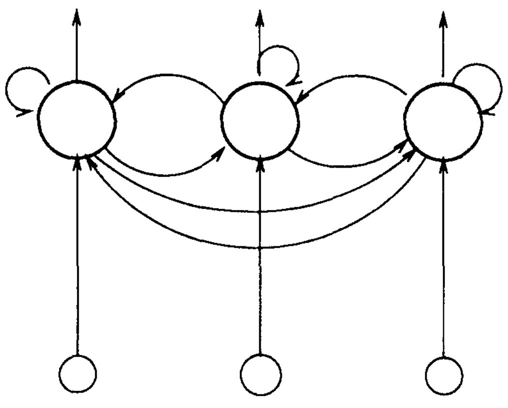
FIG. 3. Network with lateral connections.

have

$$A = \begin{cases} a_{ij} & i = j, \\ a_{ij} \leq 0 & i \neq j \end{cases}$$
$$B = B_1.$$

In addition, all elements of B here are positive. The feedback now brings the nonlinear function $g(\cdot)$ into play and so the properties of the recurrent network are richer than those of the non-recurrent network. This kind of network is useful in producing quantization of the inputs, or can serve as a local or global peak detector. The combination of several types of connections can create complex architectures like analogue Adaptive Resonance Theory (ART, and its later derivates) (Grossberg (1988)). This system is a two layer architecture with feedback and lateral interaction capable of creating different classes to classify signals according to their similarities.

The idea of encoding inputs has been used to generate nonlinear mappings. A network which uses this characteristic is the counterpropagation network (CPN) (Hecht-Nielsen (1987a)). This network has a first layer of competitive units, known as the Kohonen or LVQ layer (Kohonen (1987)), and a second layer of output units. In this case the output of the network can be regarded as a linear combination of functions that represent the characteristic of the encoder, i.e.

$$y_i = \sum_{j=1}^{N} a_{ij}\phi_j(u_j), \tag{29}$$

where ϕ_j can have different forms, one of which uses lateral inhibition.

2.5.3. *Cerebellar model articulation controller.* Another network which uses the encoding system to approximate nonlinear functions is the Cerebellar Model Articulation Controller (CMAC) (Albus (1975a, b)), even though CMAC does not use a competitive network to produce an encoding of the input to reach the same general objective (radial basis function networks and splines also fall within this framework). These networks use the concept of locally tuned overlapping receptive fields. They have the advantage that they can approximate complex nonlinear functions much faster than networks using sigmoids. Only a small subset of the parameters are adjusted at each point in the input space. This reduces sensitivity to the order in which the training data are presented. A complete and detailed account of the relationship between splines and CMAC is given in Lane *et al.* (1991).

The CMAC architecture has been applied to nonlinear control problems (Ersü and Tolle (1984)). The use of the system as an encoder not only gives the possibility of nonlinear mapping but also a fault tolerant characteristic.

2.5.4. *Cellular neural networks.* (Chua and Yang (1988a, b)) proposed an architecture derived from cellular automata (Wolfram (1986)), called the Cellular Neural Network (CNN). Each layer of a CNN is a two-dimensional structure with sparse interconnections and local properties based on spatial-invariant templates. The underlying idea was to propose a two-dimensional fast parallel image processing filter (Dzieliński *et al.* (1990)), although optimization applications can be found (*Proc. CNNA'90* (1990)). No control applications have yet been reported.

3. LEARNING IN STATIC NETWORKS

The application of artificial neural networks normally proceeds in two phases; for the architectures described in Section 2 we assumed that the networks had already been trained using suitable data. A trained network with constant coefficients then provides a fixed behaviour (in the context of associative memory this is called the recall phase). In this section (static networks) and in the following section (dynamic networks) we discuss the training phase and present algorithms for automatic adjustment of network coefficients—a process called learning in the ANN literature and adaptation in the control literature.

A learning algorithm is associated with any change in the memory as represented by the weights; learning does not in this sense imply a change in the structure of the memory (structural learning is a separate issue which is receiving some attention). From this point of view learning can be regarded as a parametric adaptation algorithm.

Learning algorithms can be classified into two main groups: supervised learning, which incorporates an external reference signal (teacher) and/or global information about the system, and unsupervised learning, which incorporates no external reference signal and relies only upon local information and internal signals.

In the following we present, in a standard and unifying fashion, a learning algorithm based on systems identification theory (Ljung and Söderström (1983)). Within this framework we show that backpropagation is a special case of the general algorithm.

3.1. *Non-dynamic single-layer architecture*

From equation (15) the single-layer non-dynamic architecture is described by:

$$\begin{aligned} x(t) &= Bu(t), \\ y(t) &= g(x(t)), \end{aligned} \tag{30}$$

where we assume $w = 0$. The standard ANN learning problem can be posed as follows:

- y and u contain signals available for measurement.
- The function g is prespecified, typically taken from Table 2.
- A desired output value, or reference value, r (corresponding to each u) is known.
- Find a parameter estimate $\hat{B}$ such that the square of the error $e(t)$ between r and y (i.e. $e^2(t) = (r(t) - y(t))^2$) is minimized over all pairs y, u.

Backpropagation, a learning algorithm common in the ANN literature, is a special case of this general problem. In the linear case when $g(x) = x$ (i.e. $y = x$) this corresponds to the standard linear in the

TABLE 3. SUMMARY OF ARCHITECTURES

Architecture	Type of connection	Main characteristic	References
Perceptron	1 layer feedforward	Linear system with on–off output	Fu (1976)
Associative Reward–Penalty	3 layers feedforward	Use of low quality information as feedback	(Barto (1988); Barto et al. (1983); Helferty et al. (1989); Anderson (1989); Porcino and Collins (1990); Guhua and Mathur (1990); Chen (1990); Franklin (1989); Lee and Berenji (1989))
Backpropagation	several layers feedforward	Nonlinear system decision regions	(Goldberg and Pearlmutter (1988); Yeung and Beckey (1989); Bassi and Beckey (1989); Josin (1988); Sekiguchi et al. (1989); Chen and Pao (1989); Psaltis et al. (1988); Guez and Selinsky (1990); Josin (1990); Grant and Zhang (1989); Sanner and Akin (1990); Chen (1989); Karsai et al. (1989); Troudet and Merril (1989); Ciliz and Isik (1989); Bhart and McAvoy (1990); Ydstie (1990); Ungar et al. (1990); Savic and Tan (1989); Bhat et al. (1990))
Cohen–Grossberg	1 layer feedback	Dynamic nonlinear system	(Guez et al. (1988); Bavarian (1988); Zhao and Mendel (1988); Pearlmutter (1988); Żak (1990); Chi et al. (1990))
LVQ	lateral connections	Quantization	(Graf and Lalonde (1988); Martinez et al. (1988); Ritter and Schulten (1986, 1988))
CPN	lateral connections feedforward	Nolinear system (look-up table)	(Miller et al. (1987); Atkenson and Reinkensmeyer (1989); Ersü and Tolle (1984); Cheok and Huang (1989); Kraft and Campagna (1990))
ART II	2 layers feedback lateral connections	Adaptive classifer (look-up table)	(Kumar and Guez (1990); Bullock (1988))
CMAC	2 feedfoward layers, overlapping receptive fields	Look-up table	(Albus (1975b); Ersü and Tolle (1984); Kraft and Campagna (1990); Miller et al. (1987))

parameters parameter identification problem. It would then typically be solved in continuous time by the equation

$$\dot{\hat{B}}(t) = P(t)u(t)[r(t) - y(t)], \qquad (31)$$

and in discrete time by

$$\hat{B}(t+1) = \hat{B}(t) + P(t)u(t)[r(t) - y(t)], \qquad (32)$$

where t in the last equation is the integer time index. Various choices of P could be used. For example:

- $P = \alpha I$, α is a scalar constant—(gradient algorithm).
- $P = \dfrac{\alpha}{\|u\|} I$, α is a scalar constant—(normalized gradient algorithm (Widrow and Lehr (1990)).
- $\dot{P}(t) = P(t)u(t)u(t)^T P(t)$, if P^{-1} is non-singular—(least-squares).

These methods increase in sophistication at the expense of increased computation. In particular the least-squares approach may be unfeasible for a large number of units N.

In the ANN literature it is very common to use the linear in the parameters approach as exemplified by equations (31) and (32) even though nonlinear functions are involved. In addition, the simple gradient algorithm is normally used. Note, however, that the neuron nonlinearity precludes simple application of linear least-squares algorithms for adaptation of neuron weights.

3.2. *Non-dynamic multiple layers*

The methods of Section 3.1 cannot be applied to the multilayer ANN of Section 2.3 as neither the input to, nor the reference signals for, the hidden layers are known. For many years this hindered this branch of ANNs until the backpropagation (BP) algorithm, developed by Werbos (1974), was rediscovered and popularized by Rumelhart *et al.* (1986). An interesting perspective on this is given by Minsky and Papert in the epilogue of the second edition of their book "Perceptrons" (Minsky and Papert (1988)).

The BP algorithm can be seen as a gradient algorithm applied to a nonlinear optimization problem. Briefly, the BP algorithm solves the missing information problem as follows:

- Inputs to the hidden layers are taken as the inputs to the first layers propagated forward through the network.
- The effective reference signals for the hidden layers are obtained by backpropagation of the error through the network. This is achieved by taking the partial derivative of the squared error against the parameters.

3.2.1. *Derivation of the backpropagation algorithm.* Backpropagation (BP) has brought to the attention of neural network researchers by the PDP Group Rumelhart and McClelland (1986). Rumelhart *et al.* (1986) rediscovered, however, the algorithm derived in an entirely different context by Werbos (1974). As outlined above, BP solves the problem of hidden layers learning, which is why the PDP contribution is widely recognised. Below we give Werbos' derviation of the algorithm, which is more general and mathematically more rigorous than the one given in Rumelhart and McClelland (1986).

Let an ordered system of equations (Werbos (1989)) be given as

$$x_i = f_i(x_{i-1}, \ldots, x_r), \quad i = 1, \ldots, N+1. \qquad (33)$$

The formally proper method of calculating the partial derivatives of x_{N+1} is given by:

$$\frac{\partial^+ x_{N+1}}{\partial x_i} = \sum_{j>i}^{N+1} \frac{\partial^+ x_{N+1}}{\partial x_j} \frac{\partial f_j}{\partial x_i}. \qquad (34)$$

Formula (34) is a recursive definition of the ordered derivative $\partial^+ x_{N+1}/\partial x_i$ for the systems (33).

A feedforward network is given a reference signal d only for the output and it is required to minimize

$$E = \tfrac{1}{2} \sum_{k=1}^{P} (d^k - x)^T (d^k - x), \tag{35}$$

which is the nonlinear least-squares fitting problem (Eykhoff (1974)) for P patterns. The gradient algorithm for adjusting weights yields

$$a_{ij}^{\text{new}} = a_{ij}^{\text{old}} - \alpha \frac{\partial E}{\partial a_{ij}}, \quad \alpha > 0, \tag{36}$$

where ($P = 1$ assumed for simplicity)

$$\frac{\partial E}{\partial a_{ij}} = \frac{\partial E}{\partial x_i}\frac{\mathrm{d}x_i}{\mathrm{d}s_i}\frac{\partial s_i}{\partial a_{ij}} = \frac{\partial E}{\partial x_i} g'(s_i) x_j. \tag{37}$$

In (37) s_i is the total input to ith neuron, i.e. a weighted sum of the outputs of neurons of the previous layer. For the output layer $\partial E/\partial x_i = x_i - d_i$, but it is not obvious how to find the error derivative for the hidden layers. The backpropagation algorithm assumes linear propagation of the error derivative

$$\frac{\partial E}{\partial x_i} = \sum_j \frac{\partial E}{\partial x_j}\frac{\mathrm{d}x_j}{\mathrm{d}s_j}\frac{\partial s_j}{\partial x_i} = \sum_j \frac{\partial E}{\partial x_j} g'(s_j) a_{ji}, \tag{38}$$

where x_i belongs to layer l, and x_j to $l + 1$. Beginning with the output layer this can be recursively solved. This formulation (Rumelhart and McClelland (1986)) requires care with neuron indexing and is imprecise as far as the definition of the partial derivative is concerned.

On the other hand, the network is an ordered system. Thus, (38) can be expressed as

$$\frac{\partial^+ E}{\partial x_i} = \frac{\partial E}{\partial x_i} + \sum_{j>i}^{N} \frac{\partial^+ E}{\partial x_j}\frac{\partial x_j}{\partial x_i}, \tag{39}$$

with E substituted for x_{N+1} in (34), and is no longer an heuristic assumption. Then the BP algorithm for feedforward networks may be written as

$$\begin{aligned} z_i &= \sum_{j>i} a_{ji} g'(s_j) z_j + \epsilon_i, \\ \frac{\partial E}{\partial a_{ij}} &= g'(s_i) z_i x_j, \\ a_{ij}^{\text{new}} &= a_{ij}^{\text{old}} - \alpha \frac{\partial E}{\partial a_{ij}}, \end{aligned} \tag{40}$$

where $z_i = \partial^+ E/\partial x_i$ and $\epsilon_i = \partial E/\partial x_i$, or in vector–matrix form:

$$\begin{aligned} \Gamma &= \operatorname{diag}[g'(s_1), \ldots, g'(s_N)], \\ z &= A^T \Gamma z + \epsilon, \\ A^{\text{new}} &= A^{\text{old}} - \alpha \Gamma z x^T. \end{aligned} \tag{41}$$

Here, neurons are labelled with consecutive numbers, irrespective of the layer structure, i.e. $x_1, x_2, \ldots, x_i, \ldots, x_N$.

The procedure can be summarized as follows:

(1) Assign random values to all a_{ij} (initialize A).

(2) Input P patterns and calculate all intermediate and output signals of the network (x_js and s_js).

(3) Apply (40) to update a_{ij}.

The forward pass encompasses steps (1)–(2) and the backward pass is given by (3).

Backpropagation is an off-line technique as the forward pass must be computed for all P before (41) is applied (backward pass).

3.2.2. *Control interpretation of backpropagation.* Consider a general network represented by a nonlinear function f and a set of parameters θ, containing the coefficients of (A, B) in equation (14), described by the following equation:

$$y(t) = f(u(t), \theta), \tag{42}$$

where $y(t)$ and $u(t)$ are the output and the input of the system. Here we consider discrete time systems having the integer time index t. The problem is, given a set of points $[y(t), u(t)]$, try to find a set of weights $\hat{\theta}$ such that the mean square error E is minimized, where

$$E = \sum_{t=1}^{T} \lambda^{T-1} \|(y(t) - f(u(t), \hat{\theta}))\|^2. \tag{43}$$

Here, λ is an exponential forgetting factor which gives different weights to different observations and T is the number of presentations of different training sets.

Applying a nonlinear estimation algorithm (Gawthrop and Sbarbaro (1990) and Albert and Gardner (1967)), the estimates $\hat{\theta}(t)$ can be calculated as

$$\hat{\theta}(t+1) = \hat{\theta}(t) + P^+(t)\bar{x}(t)[y(t) - f(u(t), \hat{\theta})], \tag{44}$$

where $\bar{x}(t)$ is the partial derivative of the output against the parameter, that is $\frac{\partial F}{\partial \hat{\theta}}$, and $P^+(t)$ is the pseudo-inverse of a matrix P given by

$$P(t) = \sum_{\tau=1}^{T} \lambda^{T-\tau} \bar{x}(\tau)\bar{x}(\tau)^T. \tag{45}$$

The backpropagation algorithm is a special case of this learning algorithm obtained by setting $P^+(t)$ to a constant diagonal matrix.

There are many variations of the algorithm for off-line as well as on-line use, and some decoupled versions to avoid inverting the information matrix $P(t)$, and to utilize the parallel structure and local information (Chen *et al.* (1990b); Scalero and Tepedelenlioglu (1990); Kollias (1989)). These algorithms are like BP with varying gain.

Some problems which must be faced are the definition of the structure of the network, the possibility of local minima due to data dependencies, and selecting the appropriate input signal such that the parameters converge to values which produce a good approximation of the system.

3.2.3. *Static networks representing temporal processing.* There are two approaches to processing temporal signals. The first and the most used is to window the input and then treat the time domain like another spatial domain. Here, tapped delay input is fed to a standard multilayer network, producing in this way a nonlinear ARMAX (NARMAX) model

(Chen *et al.* (1990b)). The second possibility is to use internal storage to maintain the current state having in this way a dynamic network. Dynamic networks are discussed in the following section.

The tapped delay input approach has two main disadvantages. Firstly, there is a hard limit as to the amount of temporal context. Secondly, these models have no inbuilt mechanism for dealing with variations in the rate of input (Robinson (1989)).

4. LEARNING IN DYNAMIC NETWORKS

Dynamic or recurrent networks are qualitatively different from feedforward ones, because their structure incorporates feedback. In general, the output of every neuron is fed back with varying gains (weights) to the inputs of all neurons (fully connected network). The architecture is inherently dynamic and usually one-layered, since its complex dynamics give it powerful representation capabilities. Other characteristics of dynamic networks will be highlighted below.

There are two basic dynamic network descriptions. The first (Hopfield (1984)) assumes separate state and output

$$\begin{aligned} T &= \operatorname{diag}[T_1, \dots, T_N], \\ T\dot{x} &= -x + Ay + u, \\ y &= g(x), \end{aligned} \tag{46}$$

while the other gives linear output equal to state

$$\begin{aligned} s &= Ay, \\ T\dot{x} &= -x + g(s) + u, \\ y &= x, \end{aligned} \tag{47}$$

or, in discrete time,

$$\begin{aligned} s(t) &= Ay(t), \\ Tx(t+1) &= -x(t) + g(s(t)) + u(t), \\ y(t) &= x(t). \end{aligned} \tag{48}$$

In this section we shall use equations (47)–(48). For the sake of simplicity we assume T to be the identity matrix ($T = I$), unless otherwise stated.

4.1. *Dynamic behaviour learning*

Recurrent networks possessing the same structure can exhibit different dynamic behaviour, due to the use of distinct learning algorithms. The network is defined when its architecture and learning rule are given. In other words, it is a composition of two dynamic systems: transmission and adjusting systems. The overall input–output behaviour is thus a result of the interaction of both.

There are two general concepts of recurrent structures training. Fixed point learning is aimed at making the network reach the prescribed equilibria or perform steady-state matching. The only requirement on the transients is that they die out. Trajectory learning trains the network to follow the desired trajectories in time. In particular, when $t \to \infty$, it will also reach the prescribed steady-state, so it can be viewed as a generalization of fixed point algorithms.

4.2. *Fixed point learning*

For the network (47) the error is defined as

$$E = \tfrac{1}{2}\sum_{k=1}^{P} (y_\infty^k - y)^T (y_\infty^k - y), \tag{49}$$

where y_∞^k are vectors of the desired equilibria, being the solutions of (47) with the left-hand side set to 0:

$$0 = -y_\infty^k + g(s) + u, \quad k = 1, \dots, P. \tag{50}$$

Comparison of (49) with (35) shows that similar error functions are minimized. However, d^k in (35) do not affect network's dynamics (because it has none) and y_∞^k in (49) define with (50) the steady state of a dynamic network.

During learning the network does not receive any external inputs. It is excited by initial conditions corresponding to the expected workspace and evolves with y_∞ as a constant reference signal, which is called relaxation (Pineda (1989)). Practically, this is done using recurrent backpropagation (Pineda (1987); Almeida (1988)) (formulae given assume $P = 1$ in (49) for simplicity):

$$\begin{aligned} e_i &= \begin{cases} y_{\infty_i} - y_i, & \text{if } y_i \text{ is a network output;} \\ 0, & \text{otherwise,} \end{cases} \\ \dot{z}_i &= -z_i + \sum_j a_{ji} g'(s_j) z_j + e_i, \\ \dot{a}_{ij} &= \alpha g'(s_i(\infty)) z_i(\infty) y_i(\infty), \end{aligned} \tag{51}$$

or, in vector–matrix form

$$\begin{aligned} e &= [0 \vdots y_\infty - y]^T, \\ \Gamma &= \operatorname{diag}[g'(s_1), \dots, g'(s_n)], \\ \dot{z} &= -z + A^T \Gamma z + e, \\ \dot{A} &= \alpha \Gamma(\infty) z(\infty) y^T(\infty). \end{aligned} \tag{52}$$

Since only $\lim_{t\to\infty} z(t) = z(\infty)$ is of interest, the initial condition for the differential equation for z in (52) can theoretically be arbitrary. Practically, numerical soundness should be borne in mind.

The idea exploited in (50) is used to calculate z, which corresponds to the ordered derivative (34); also e_i from (51) is equal to $-\epsilon_i$ of (40) (see Section 3.2.1). The equation for A can be solved after the network has settled, since $y(\infty)$ and $z(\infty)$ (steady-state values of $y(t)$ and $z(t)$, respectively) are needed.

This algorithm differs from Hopfield nets (Hopfield (1982, 1984)) (compare Section 2.4.1), since there the learning was performed by the Hebbian rule (cf. (22)) and the relaxation is done only during the recall phase (the Hopfield net is a content-addressable memory). Recurrent backpropagation creates basins of attraction around the given equilibria by adjusting the weights, learning them through relaxation (52). The Hopfield net generates its attractors and basins in applying the Hebbian rule and thus produces equilibria on its own, representing input patterns.

If the forward pass in recurrent backpropagation is stable, so is the backward one (Almeida (1988)). On the other hand, the Hopfield net should be stable, because defining it requires guessing a Lyapunov function appropriate to an application, (e.g. Park and Lee (1990)).

4.3. *Trajectory learning*

In this case, the error for the network (47) is given by

$$E = \tfrac{1}{2}\int_{t_0}^{t_1} [(d(\tau) - y(\tau))^T(d(\tau) - y(\tau))]\,\mathrm{d}\tau, \quad (53)$$

where $d(\tau)$ is a vector of the desired trajectory and t_1 may be a constant (off-line techniques) or a variable (on-line algorithms). The discrete-time version (see (48)) yields:

$$E = \tfrac{1}{2}\sum_{\tau=t_0}^{t_1} [(d(\tau) - y(\tau))^T(d(\tau) - y(\tau))], \quad (54)$$

with the previous remarks valid. Both (53) and (54) may include extra summation to cover multiple trajectories $d^k(\tau)$. For simplicity, this is omitted in the following discussion.

4.3.1. *Backpropagation through time.* The simplest method, already mentioned in Rumelhart and McClelland (1986), is to unfold the network through time, i.e. replace a one-layer recurrent network with a feedforward one with t_1 layers. The equivalent static network has as many layers as time instants, which is easily understood in the discrete time context and therefore we start with this. Standard backpropagation will be applied (see Section 3.2.1), and since each layer contains the same weights "seen" in different moments, their final value is the sum of intermediate values (Werbos (1990)). Thus, we have

$$\begin{aligned}
&E = \tfrac{1}{2}\sum_{\tau=t_0}^{t_1}\sum_i (d_i(\tau) - y_i(\tau))^2, \quad t_1 = \text{const},\\
&e_i(\tau) = \begin{cases} d_i(\tau) - y_i(\tau), & \text{if } y_i(\tau) \text{ is a network output;}\\ 0, & \text{otherwise,}\end{cases}\\
&z_i(\tau) = \sum_j a_{ji} g'(s_j(\tau+1)) z_j(\tau+1) - e_i(\tau), \quad z_i(t_1) = 0,\\
&\frac{\partial E}{\partial a_{ij}} = \sum_{\tau=t_0}^{t_1} g'(s_i(\tau)) z_i(\tau) y_j(\tau),\\
&a_{ij}^{t_1} = a_{ij}^{t_0} - \alpha\frac{\partial E}{\partial a_{ij}},
\end{aligned} \quad (55)$$

or, in vector–matrix form,

$$\begin{aligned}
&E = \tfrac{1}{2}\sum_{\tau=t_0}^{t_1} [(d(\tau) - y(\tau))^T(d(\tau) - y(\tau))],\\
&t_1 = \text{const},\\
&e(\tau) = [0 \,\vdots\, d(\tau) - y(\tau)]^T,\\
&\Gamma(\tau) = \operatorname{diag}[g'(s_1(\tau)), \ldots, g'(s_n(\tau))],\\
&z(\tau) = A^T(t_0)\Gamma(\tau+1)z(\tau+1) - e(\tau), \quad z(t_1) = 0,\\
&\nabla_A E = \sum_{\tau=t_0}^{t_1} \Gamma(\tau) z(\tau) y^T(\tau),\\
&A^{t_1} = A^{t_0} - \alpha\nabla_A E.
\end{aligned} \quad (56)$$

The definition of $e_i = e_i(\tau)$ in (55) and/or $e = e(\tau)$ in (56) will be further used without additional explanation.

Theoretically, there exists the possibility (Williams and Zipser (1990)) of an on-line algorithm, but the above calculations would have to be performed at every step, requiring unlimited memory and computational power.

The continuous time version was introduced heuristically by Pearlmutter (1989), and then rediscovered by Sato (1990), whose derivation, based on the calculus of variations, is mathematically rigorous and will be briefly presented here. Introduce the Lagrangian

$$L = \int_{t_0}^{t_1} \Big\{\tfrac{1}{2}\sum_i (d_i - y_i)^2 - \sum_i z_i[T_i\dot{y}_i + y_i - g(s_i) - u_i]\Big\}\,\mathrm{d}\tau, \quad (57)$$

where z_i are Lagrange multipliers (note that we show a derivation for $T \neq I$; see remarks after (47)–(48)). The first variation yields

$$\begin{aligned}
\delta L = \int_{t_0}^{t_1} \Big\{&\sum_i [(y_i - d_i) - z_i]\,\delta y_i\\
&+ \sum_i z_i g'(s_i)\sum_j a_{ij}\,\delta y_i - \sum_i z_i T_i\,\delta\dot{y}_i\\
&+ \sum_i z_i g'(s_i)\sum_j y_j\,\delta a_{ij}\Big\}\,\mathrm{d}\tau,
\end{aligned} \quad (58)$$

which can be reduced by defining the auxiliary equation

$$T_i\dot{z}_i = z_i - \sum_j a_{ji} g'(s_j) z_j - e_i. \quad (59)$$

Multiplying both sides of (59) by δy_i and summing with respect to i gives

$$\delta L = \sum_i [T_i z_i(t_0)\,\delta y_i(t_0) - T_i z_i(t_1)\,\delta y_i(t_1)] + \int_{t_0}^{t_1}\Big[\sum_i z_i g'(s_i)\sum_j y_j\,\delta a_{ij}\Big]\,\mathrm{d}\tau. \quad (60)$$

Since the initial conditions $y_i(t_0)$ do not depend on weights, $\delta y_i(t_0) = 0$. If additionally the boundary values

$$z_i(t_1) = 0, \quad (61)$$

are imposed then

$$\delta L = \int_{t_0}^{t_1}\Big[\sum_i z_i g'(s_i)\sum_j y_j\,\delta a_{ij}\Big]\,\mathrm{d}\tau, \quad (62)$$

and hence (compare (55))

$$\frac{\partial E}{\partial a_{ij}} = \int_{t_0}^{t_1} g'(s_i) z_i y_j\,\mathrm{d}\tau, \quad (63)$$

$$\dot{a}_{ij} = -\alpha\frac{\partial E}{\partial a_{ij}}, \quad (64)$$

which together with (59) completes the algorithm. The vector–matrix form looks as follows:

$$\begin{aligned}
&E = \tfrac{1}{2}\int_{t_0}^{t_1} [(d(\tau) - y(\tau))^T(d(\tau) - y(\tau))]\,\mathrm{d}\tau,\\
&t_1 = \text{const},\\
&T\dot{z} = z - A^T\Gamma z - e,\\
&z(t_1) = 0,\\
&\nabla_A E = \int_{t_0}^{t_1} \Gamma(\tau) z(\tau) y^T(\tau)\,\mathrm{d}\tau,\\
&\dot{A} = -\alpha\nabla_A E.
\end{aligned} \quad (65)$$

Notice that the auxiliary equation (59) is defined on $[t_0, t_1]$ with $z_i(t_1) = 0$ (see (61)), which means that it must be integrated backwards in time. The method for adjusting time constants and delays can be found in Pearlmutter (1990). A speeding up modification of the algorithm was proposed in Fang and Sejnowski (1990).

The method is essentially non-recurrent, because it uses a feedforward equivalent for learning, so there is no explicit use of the network's feedback. It requires recording of the whole trajectories and playing them back to calculate the weights. However, once the weights are established the network runs as a recurrent one.

4.3.2. *Forward propagation.* Backpropagation through time was inherently an off-line technique. The following algorithm (Robinson and Fallside (1987); Williams and Zipser (1989b)) overcomes the difficulty. For the recurrent network (T_k suppressed for clarity; compare (47))

$$\dot{y}_k = -y_k + g(s_k) + u_k, \tag{66}$$

with

$$\frac{\partial \dot{y}_k}{\partial a_{ij}} = -\frac{\partial y_k}{\partial a_{ij}} + g'(s_k)\left[\delta_{ik} y_i + \sum_l a_{kl} \frac{\partial y_l}{\partial a_{ij}}\right], \tag{67}$$

where δ_{ik} is the Kronecker symbol, define error

$$E = \int_{t_0}^{t} \left[\tfrac{1}{2} \sum_k (d_k(\tau) - y_k(\tau))^2\right] d\tau, \quad t \to \infty, \tag{68}$$

i.e. with free termination time (t is a variable). The rationale for introducing the subscript k in (66) is that each neuron is fully connected, so it is influenced by all weights a_{ij} (cf. (67)).

Consider the first variation of the error

$$\delta E = \sum_k \left[\int_{t_0}^{t} [e_k(\tau)\, \delta y_k]\, d\tau\right], \tag{69}$$

where $e_k(\tau)$ is defined as in (55). Since a_{ij}s are independent variables, the following holds:

$$\delta y_k = \sum_i \sum_j \frac{\partial y_k}{\partial a_{ij}} \delta a_{ij}. \tag{70}$$

Taking into account constraints (66) gives, with $p_{ij}^k = \partial y_k / \partial a_{ij}$ substituted to (67),

$$\frac{\partial E}{\partial a_{ij}} = \sum_k \left[\int_{t_0}^{t} [e_k(\tau) p_{ij}^k(\tau)]\, d\tau\right],$$

$$\dot{p}_{ij}^k = -p_{ij}^k + g'(s_k)\left[\delta_{ik} y_i + \sum_l a_{kl} p_{ij}^l\right], \tag{71}$$

$$\dot{a}_{ij} = -\alpha \frac{\partial E}{\partial a_{ij}},$$

and

$$p_{ij}^k(t_0) = 0, \tag{72}$$

because initial conditions, $y_k(t_0)$, do not depend on weights. The replacement of the boundary values (61) with the initial conditions (72) allows on-line calculations in (71). It is not straightforward to write down (71) and (72) in the vector–matrix form, because variables p_{ij}^k require a three-dimensional array. One possibility is:

$$\begin{aligned} E &= \tfrac{1}{2} \int_{t_0}^{t} [(d(\tau) - y(\tau))^T (d(\tau) - y(\tau))]\, d\tau, \\ t &\to \infty, \\ \nabla_A E &= \sum_k \left[\int_{t_0}^{t} [e_k(\tau) P_k(\tau)]\, d\tau\right], \\ \dot{P}_k &= -P_k + g'(s_k)[\lambda^k y^T + \Pi_k], \quad k = 1, \ldots, N, \\ P_k(t_0) &= 0, \quad k = 1, \ldots, N, \\ \dot{A} &= -\alpha \nabla_A E, \end{aligned} \tag{73}$$

where $P_k = [p_{ij}^k]$ and $\Pi_k = [\pi_{ij}]$, $\pi_{ij} = \sum_l a_{kl} p_{ij}^l$ are square matrices of dimension N and $\lambda^k = [0, \ldots, \underbrace{1}_{\lambda_k}, \ldots, 0]^T$ is a vector.

A modification of the method can be found in Zipser (1989).

Forward propagation, called also real-time recurrent learning (RTRL) (Williams and Zipser (1989a)), requires non-local computations, since to calculate $\partial E / \partial a_{ij}$ knowledge of all p_{ij}^k is needed (see (71)). Real-time solving of the auxiliary equation on p_{ij}^k contributes to computing complexity, but the algorithm is truly on-line.

It is worth noting that recurrent networks allow an elegant solution of the key problem of multilayer feedforward networks, viz. the reference signal for hidden units. The answer is straightforward (see (51)) by setting the error to 0. It does not make impossible calculation of weights for hidden units (compare (71)) since through feedback they get information about the system performance.

4.3.3. *Trajectory learning: identification and optimal control.* Most of the problems of neural networks learning can be expressed in terms of an index minimization, which is usually a squared error function. A typical solution is an application of the gradient algorithm whose convergence properties are quite poor (Fletcher (1987)). However, the problem involved is not a classical optimization task, because the function to be minimized, $E(A)$, is not given explicitly. Thus, an N^2-component gradient (A is an $N \times N$ matrix) must be estimated which explains why algorithms making use of the second-order information (Newton methods exploiting Hessian-related data) are not developed despite their quadratic rate of convergence.

Dynamic neural networks are parametric models, which raises the question of their relation to identification. It should be borne in mind that the systems (47)–(48) are nonlinear state-space models. In the theory of nonlinear identification there exist several methods for certain special cases (Haber and Unbehauen (1990)). However, they are limited to particular input–output models. Hence dynamic networks can be seen as an alternative approach of capacity that remain to be investigated.

5. REPRESENTATION, IDENTIFICATION AND CONTROL STRUCTURES

In this section we review the possibilities for using neural networks directly in nonlinear control strategies. In this context neural networks are viewed as a process modelling formalism, or even a knowledge representation framework; our knowledge about the plant dynamics and mapping characteristics is implicitly stored within the network.

The ability of networks to approximate nonlinear mappings is thus of central importance in this task. For this reason we first review a body of theoretical work which has characterized, from an approximation theory viewpoint, the possibilities of nonlinear functional approximation using neural networks.

Second, we discuss learning structures from training networks to represent forward and inverse dynamics of nonlinear systems. Finally, a number of control structures in which such models play a central role are reviewed. These established control structures provide a basis for nonlinear control using neural networks.

In the same way that transfer functions provide a generic representation for linear black-box models, ANNs potentially provide a generic representation for nonlinear black-box models.

5.1. *Networks, approximation and modelling*

The nonlinear functional mapping properties of neural networks are central to their use in control. Training a neural network using input–output data from a nonlinear plant can be considered as a nonlinear functional approximation problem. Approximation theory is a classical field of mathematics; from the famous Weierstrass Theorem (Burkill and Burkill (1970); Rudin (1976)) it is known that polynomials, and many other approximation schemes, can approximate arbitrarily well a continuous function. Recently, considerable effort has gone into the application of a similar mathematical machinery in the investigation of the approximation capabilities of networks.

A number of results have been published showing that a feedforward network of the multilayer perceptron type can approximate arbitrarily well a continuous function (Cybenko (1988, 1989); Funahashi (1989); Hornik *et al.* (1989); Carrol and Dickinson (1989)). To be specific, these papers prove that a continuous function can be arbitrarily well approximated by a feedforward network with only a single internal hidden layer, where each unit in the hidden layer has a continuous sigmoidal nonlinearity. The use of nonlinearities other than sigmoids is also discussed.

These results provide no special motivation for the use of networks in preference to, say, polynomial methods for approximation; both approaches share the "Weierstrass Property". Such comparitive judgements must be made on the basis of issues such as parsimony. Namely, do networks or polynomials require fewer parameters? For network approximators, key questions are how many layers of hidden units should be used, and how many units are required in each layer? Of course, the implementation characteristics of approximation schemes also provide a further basis for comparison (for example, the parallel, distributed nature of networks is important). For the moment, however, we concentrate purely on approximation properties.

Although the results referred to above at first sight appear attractive they do not provide much insight into these practical questions. In fact, the degree of arbitrariness of the approximation achieved using a sigmoidal network with one hidden layer is reflected in a corresponding arbitrariness in the number of units required in the hidden layer; the results were achieved by placing no restriction on the number of units used. Cybenko himself says (Cybenko (1989)) "we suspect quite strongly that the overwhelming majority of approximation problems will require astronomical numbers of terms".

What is needed now is an indication of the numbers of layers/units required to achieve a specific degree of accuracy for the function being approximated. Some work along these lines is given in Chester (1990). That paper gives theoretical support to the empirical observation that networks with two hidden layers appear to provide higher accuracy and better generalization than a single hidden layer network, and at a lower cost (i.e. fewer total processing units). Guidelines derived from a mix of theoretical and heuristic considerations are given in the introductory paper by Lippmann (1987).

From the theoretical point of view the work of Kolmogorov (1957) (see also Lorentz (1976))) did appear to throw some light on the problem of exact approximation (Poggio (1982); Hecht-Nielsen (1987b); Lippmann (1987)). Kolmogorov's theorem (a negative resolution of Hilbert's 13th problem) states that any continuous function of N variables can be computed using only linear summations and nonlinear but continuously increasing functions of only one variable. In the network context the theorem can be interpreted as explicitly stating that to approximate any continuous function of N variables requires a network having $N(2N + 1)$ units in a first hidden layer and $(2N + 1)$ units in a second hidden layer. However, it has recently been pointed out (Hornik *et al.* (1989); Girosi and Poggio (1989)) that the practical value of this result is tenuous for a number of reasons:

(1) Kolmogorov's Theorem requires a different nonlinear processing function for each unit in the network.
(2) The functions in the first hidden layer are required to be highly non-smooth. In practice this would lead to problems with generalization and noise robustness.
(3) The functions in the second hidden layer depend upon the function being approximated.

It is apparent that these conditions may be violated in the practical situations of interest here. However, a well motivated dissenting viewpoint to that presented in Girosi and Poggio (1989) has been put forward by Kurkova (1991). In that work it is argued that the above points are nullified if one is only trying to approximate a function. We leave it to the reader to explore these conflicting viewpoints further in this rapidly developing area.

From the foregoing discussion it is clear that the property of approximating functions arbitrarily well is not sufficient for characterizing good approximation schemes (since many schemes have this property), nor does this property help in justifying the use of one particular approximation scheme in preference to another. This observation has been deeply considered by Girosi and Poggio (1990) (see also the expository paper (Poggio and Girosi (1990)). These authors propose that the key property is not that of arbitrary approximation, but the property of *best approximation*. An approximation scheme is said to have this property if in the set of approximating functions there is one which has the minimum distance from the given function (a precise mathematical formulation is given in the paper). The first main result of their paper (Girosi and Poggio (1990)) is that multilayer perceptron networks do not have the best approximation property. Secondly, they prove that Radial Basin Function networks (for example, Gaussian networks) do have the best approximation property. Thus, although one must bear in mind the precise mathematical formulation of "best", there is theoretical support for favouring RBF networks. Moreover, these networks may always be structured with only a single hidden layer and trained using linear optimization techniques with a guaranteed global solution (Broomhead and Lowe (1988)).

As with sigmoidal feedforward networks, however, there remain open questions regarding the network complexity required (i.e. the number of units). For RBF networks this question is directly related to the size of the training data. A related issue is that of choosing the centres of the basis functions (for results on automatically selecting the centres (see Moody and Darken (1989); Chen *et al.* (1990a)). Girosi and Poggio (1990) and Broomhead and Lowe (1988) discuss methods for achieving almost best approximation using restricted complexity RBF networks. This is discussed further in Sbarbaro and Gawthrop (1991).

Thus, although RBF networks have the best approximation property, it is necessary to consider the cost of utilizing this approach. For high-dimensional input spaces the number of nodes needed grows with the dimension of the input space (Hartman and Keeler (1992); Weigand *et al.* (1990)). This is in contrast to sigmoids which slice up the space into feature regions and hence are more economical with the number of nodes. This belief is rigorously supported by recent results of Barron (1991) which show that members of a certain class of "smooth" functions can be approximated by a two-layer network and that the error between the target function and the trained network can be bounded in terms of the network's size. It is shown in Barron (1991) that the error grows linearly with the dimension of the input space.

In summary, a large body of theoretical results relating to approximation using neural networks exist. These results provide theoretically important possibility theorems and deep insight into the ultimate performance of networks. They are not constructive results which define the type and structure of a suitable network for a given problem. Restraint must therefore be exercised when citing these results in support of the practical application of neural networks.

5.2. *Identification*

Although a number of key theoretical problems remain, the results discussed above do demonstrate that neural networks have great promise in the modelling of nonlinear systems. Without reference to any particular network structure we now discuss architectures for training networks to represent nonlinear dynamical systems and their inverses.

An important question in system identification is that of system identifiability (see Ljung and Söderström (1983), Ljung (1987), and Söderström and Stoica (1989)) i.e. given a particular model structure, can the system under study be adequately represented within that structure? In the absence of such concrete theoretical results for neural networks we proceed under the assumption that all systems we are likely to study belong to the class of systems that the chosen network is able to represent.

5.2.1. *Forward modelling*. The procedure of training a neural network to represent the forward dynamics of a system will be referred to as forward modelling. A structure for achieving this is shown schematically in Fig. 4. The neural network model is placed in parallel with the system and the error between the system and network outputs (the prediction error) is used as the network training signal. As pointed out by Jordan and Rumelhart (1991) this learning structure is a classical supervised learning problem where the teacher (i.e. the system) provides target values (i.e. its outputs) directly in the output coordinate system of the learner (i.e. the network model). In the particular case of a multilayer perceptron type network straightforward back-propagation of the prediction error through the network would provide a possible training algorithm.

An issue in the context of control is the dynamic nature of the systems under study. One possibility is to introduce dynamics into the network itself. This can be done either using recurrent networks (as discussed in a previous section) or by introducing dynamic behaviour into the neurons (see Willis *et al.* (1991)). A straightforward approach, and the one which for purposes of exposition will be followed here, is to augment the network input with signals corresponding to past inputs and outputs (this is also discussed in Section 3.2.3).

We assume that the system is governed by the following nonlinear discrete-time difference equation:

$$y^p(t+1) = f[y^p(t), \ldots, y^p(t-n+1); \quad u(t), \ldots, u(t-m+1)]. \qquad (74)$$

Thus, the system output y^p at time $t+1$ depends (in the sense defined by the nonlinear map f) on the past n output values and on the past m values of the input u. We concentrate here on the dynamical part of the system response; the model does not explicitly represent plant disturbances (for a method of including disturbances (see Chen *et al.* (1990a)). Special cases of the model (74) have been considered

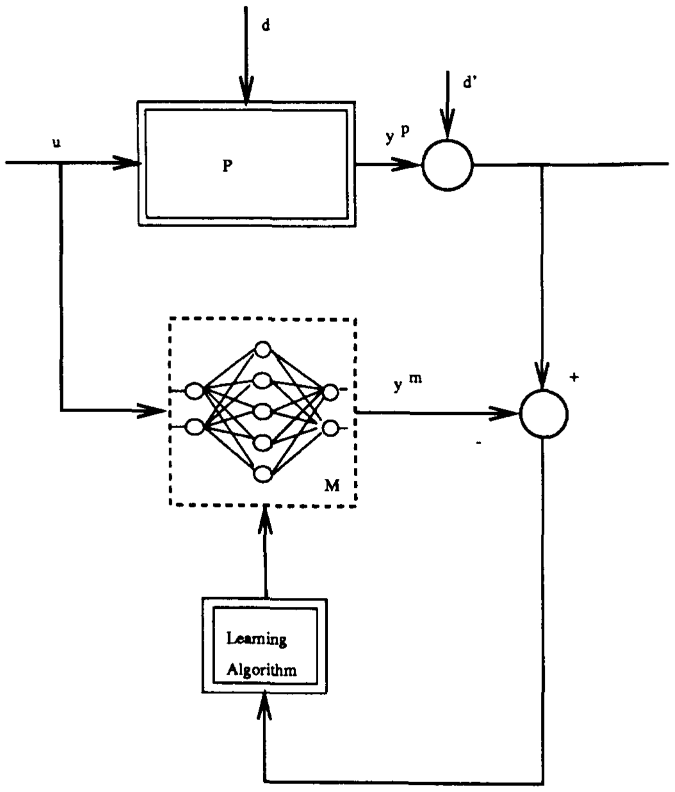

FIG. 4. Identification.

by Narendra and Parthasarathy (1990) (see also Narendra (1990)). These authors consider particular cases where the system output is linear in either the past values of y^p or u.

An obvious approach for system modelling is to choose the input–output structure of the neural network to be the same as that of the system. Denoting the output of the network as y^m we then have

$$y^m(t+1) = \hat{f}[y^p(t), \ldots, y^p(t-n+1); \quad u(t), \ldots, u(t-m+1)]. \quad (75)$$

Here, $\hat{f}$ represents the nonlinear input–output map of the network (i.e. the approximation of f). Notice that the input to the network includes the past values of the real system output (the network has no feedback). This dependence on the system output is not included in the schematic of Fig. 4 for simplicity. If we assume that after a suitable training period the network gives a good representation of the plant (i.e. $y^m \approx y^p$) then for subsequent post-training purposes the network output itself (and its delayed values) can be fed-back and used as part of the network input. In this way the network can be used independently of the plant. Such a network model is described by

$$y^m(t+1) = \hat{f}[y^m(t), \ldots, y^m(t-n+1); \quad u(t), \ldots, u(t-m+1)]. \quad (76)$$

The structure in (76) may also be used for training the network. This possibility has been discussed by Narendra (Narendra and Parthasarathy (1990); Narendra (1990)).

In the context of the identification of linear time invariant systems the two possibilities have been extensively considered by Narendra and Annaswamy (1989). The two structures have also been discussed in the signal processing literature (see Widrow and Stearns (1985)). The structure of equation (75) (referred to as the series–parallel model by Narendra) is supported in the identification context by stability results. On the other hand, (76) (referred to by Narendra as the parallel model) may be preferred when dealing with noisy systems since it avoids problems of bias caused by noise on the real system output (Widrow and Stearns (1985); Widrow (1986)).

5.2.2. *Inverse modelling*. Inverse models of dynamical systems play a crucial role in a range of control structures. This will become apparent in Section 5.3. However, obtaining inverse models raises several important issues which will be discussed.

Conceptually the simplest approach is direct inverse modelling as shown schematically in part (a) of Fig. 5 (this structure has also been referred to as generalized inverse learning (Psaltis *et al.* (1988)). Here, a synthetic training signal is introduced to the system. The system output is then used as input to the network. The network output is compared with the training signal (the system input) and this error is used to train the network. This structure will clearly tend to force the network to represent the inverse of the plant. However, there are drawbacks to this approach:

- The learning procedure is not "goal directed" (Jordan and Rumelhart (1991)); the training signal must be chosen to sample over a wide range of system inputs, and the actual operational inputs may be hard to define *a priori*. The actual

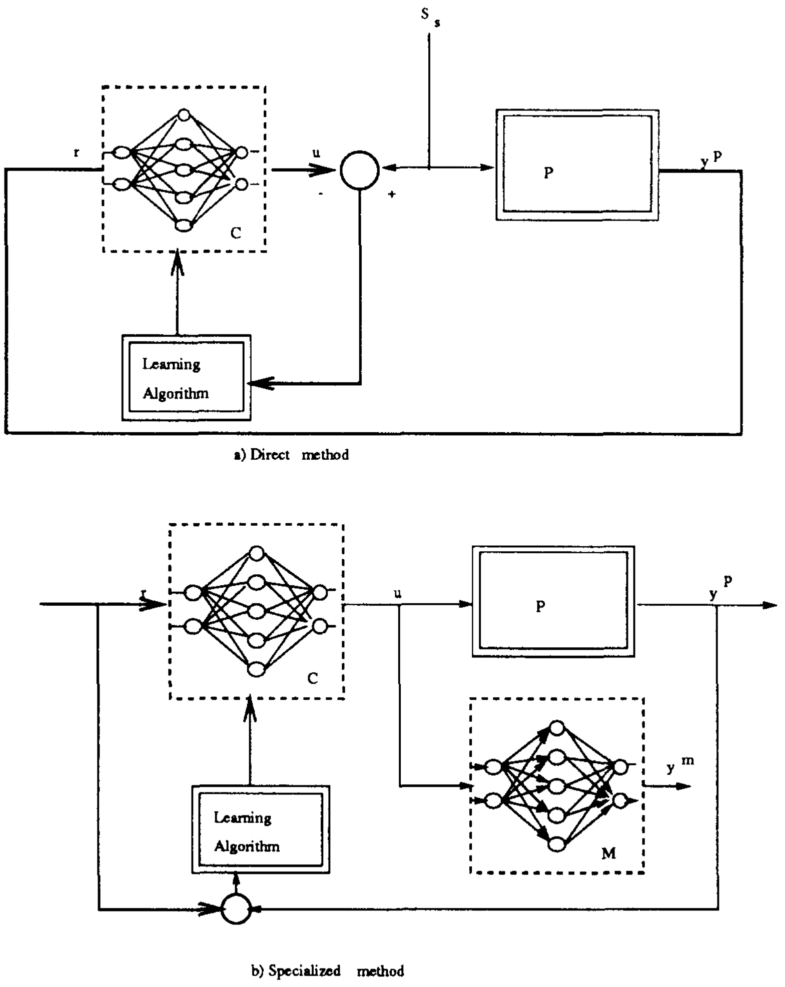

FIG. 5. Structures for inverse identification.

goal in the control context is to make the system output behave in a desired way, and thus the training signal in direct inverse modelling does not correspond to the explicit goal.

- Second, if the nonlinear system mapping is not one–one then an incorrect inverse can be obtained.

The first point above is strongly related with the general concept of persistent excitation; the importance of the inputs used to train learning systems is widely appreciated. In the adaptive control literature conditions for ensuring persistent excitation persistent excitation which will result in parameter convergence are well established (see, for example, Åström and Wittenmark (1989) and the references therein). For neural networks, methods of characterizing persistent excitation are highly desirable. A preliminary discussion on this question can be found in Narendra (1990).

A second approach to inverse modelling which aims to overcome these problems is known as specialized inverse learning (Psaltis et al. (1988)) (somewhat confusingly, Jordan and Rumelhart (1991) refer to this structure as forward modelling). The specialized inverse learning structure is shown in part (b) of Fig. 5. In this approach the network inverse model precedes the system and receives as input a training signal which spans the desired operational output space of the controlled system (i.e. it corresponds to the system reference or command signal). This learning structure also contains a trained forward model of the system (for example, a network trained as described in Section 5.2.1) placed in parallel with the plant. The error signal for the training algorithm in this case is the difference between the training signal and the system output (it may also be the difference between the training signal and the forward model output in the case of noisy systems; this obviates the need for the real system in the training procedure which is important in situations where using the real system is not viable). Jordan and Rumelhart (1991) show that using the real system output can produce an exact inverse even when the forward model is inexact; this is not the case when the forward model output is used. The error may then be propagated back through the forward model and then the inverse model; only the inverse network model weights are adjusted during this procedure. Thus, the procedure is effectively directed at learning an identity mapping across the inverse model and the forward model; the inverse model is learned as a side effect (Jordan and Rumelhart (1991)).

In comparison with direct inverse modelling the specialized inverse learning approach has the following features:

- The procedure is goal directed since it is based on the error between desired system outputs and actual outputs. In other words, the system receives inputs during training which correspond to the actual operational inputs it will subsequently receive.
- In cases in which the system forward mapping is not one–one a particular inverse will be found (Jordan and Rumelhart (1991) discuss ways in which learning can be biased to find particular inverse models with desired properties).

We now consider the input–output structure of the network modelling the system inverse. From equation (74) the inverse function f^{-1} leading to the generation of $u(t)$ would require knowledge of the future value $y^p(t+1)$. To overcome this problem we replace this future value with the value $r(t+1)$ which we assume is available at time t. This is a reasonable assumption since r is typically related to the reference signal which is normally known one step ahead. Thus, the nonlinear input–output relation of the network modelling the plant inverse is

$$u(t) = \widehat{f^{-1}}[y^p(t), \ldots, y^p(t-n+1), r(t+1); \\ u(t-1), \ldots, u(t-m+1)], \quad (77)$$

i.e. the inverse model network receives as inputs the current and past system outputs, the training (reference) signal, and the past values of the system input. In case where it is desirable to train the inverse without the real system (as discussed above) the values of y^p in the above relation are simply replaced by the forward model outputs y^m.

5.3. *Control structures*

Models of dynamic systems and their inverses have immediate utility for control. In the control literature a number of well-established and deeply analysed structures for the control of nonlinear systems exist; we focus on those structures having a direct reliance on system forward and inverse models. We assume that such models are available in the form of neural networks which have been trained using the techniques outlined above.

In the literature on neural network architectures for control a large number of control structures have been proposed and used; it is beyond the scope of this work to provide a full survey of all architectures used. In the sequel we give particular emphasis to those structures which, from the mainstream control theory viewpoint, are well-established and whose properties have been deeply analysed. First, we briefly discuss two direct approaches to control: supervised control and direct inverse control.

5.3.1. *Supervised control.* There are many control situations where a human provides the feedback control actions for a particular task and where it has proven difficult to design an automatic controller using standard control techniques (e.g. it may be impossible to obtain an analytical model of the controlled system). In some situations it may be desirable to design an automatic controller which mimics the action of the human (this has been called supervised control (Werbos (1990)).

A neural network provides one possibility for this (as an alternative approach expert systems can be used to provide the knowledge representation and control formalisms). Training the network is similar in principle to learning a system forward model as described above. In this case, however, the network input corresponds to the sensory input information received by the human. The network target outputs used for training correspond to the human control input to the system. This approach has been used in the standard pole-cart control problem (Grant and Zhang (1989)), among others.

5.3.2. *Direct inverse control.* Direct inverse control utilizes an inverse system model. The inverse model is simply cascaded with the controlled system in order that the composed system results in an identity mapping between desired response (i.e. the network inputs) and the controlled system output. Thus, the network acts directly as the controller in such a configuration. Direct inverse control is common in robotics applications; the compilation book (Miller *et al.* (1990)) provides a number of examples.

Clearly, this approach relies heavily on the fidelity of the inverse model used as the controller. For general purpose use serious questions arise regarding the robustness of direct inverse control. This lack of robustness can be attributed primarily to the absence of feedback. This problem can overcome to some extent by using on-line learning: the parameters of the inverse model can be adjusted on-line.

5.3.3. *Model reference control.* Here, the desired performance of the closed-loop system is specified through a stable reference model M, which is defined by its input–output pair $\{r(t), y^r(t)\}$. The control system attempts to make the plant output $y^p(t)$ match the reference model output asymptotically, i.e.

$$\lim_{t\to\infty} \|y^r(t) - y^p(t)\| \leq \epsilon,$$

for some specified constant $\epsilon \geq 0$. The model reference control structure for nonlinear systems utilizing connectionist models is shown in Fig. 6 (Narendra and Parthasarathy (1990)). In this structure the error defined above is used to train the network acting as the controller. Clearly, this approach is related to the training of inverse plant models as outlined above. In the case when the reference model is the identity mapping the two approaches coincide. In general, the training procedure will force the controller to be a "detuned" inverse, in a sense defined by the reference model. Previous and similar work in this area has been considering linear in the control structures (Kosikov and Kurdyukov (1987)).

5.3.4. *Internal Model Control.* In Internal Model Control (IMC) the role of system forward and inverse models is emphasised (Garcia and Morari (1982)). In this structure a system forward and inverse model are used directly as elements within the feedback loop. IMC has been thoroughly examined and shown to yield transparently to robustness and stability analysis (Morari and Zafiriou (1989)). Moreover, IMC extends

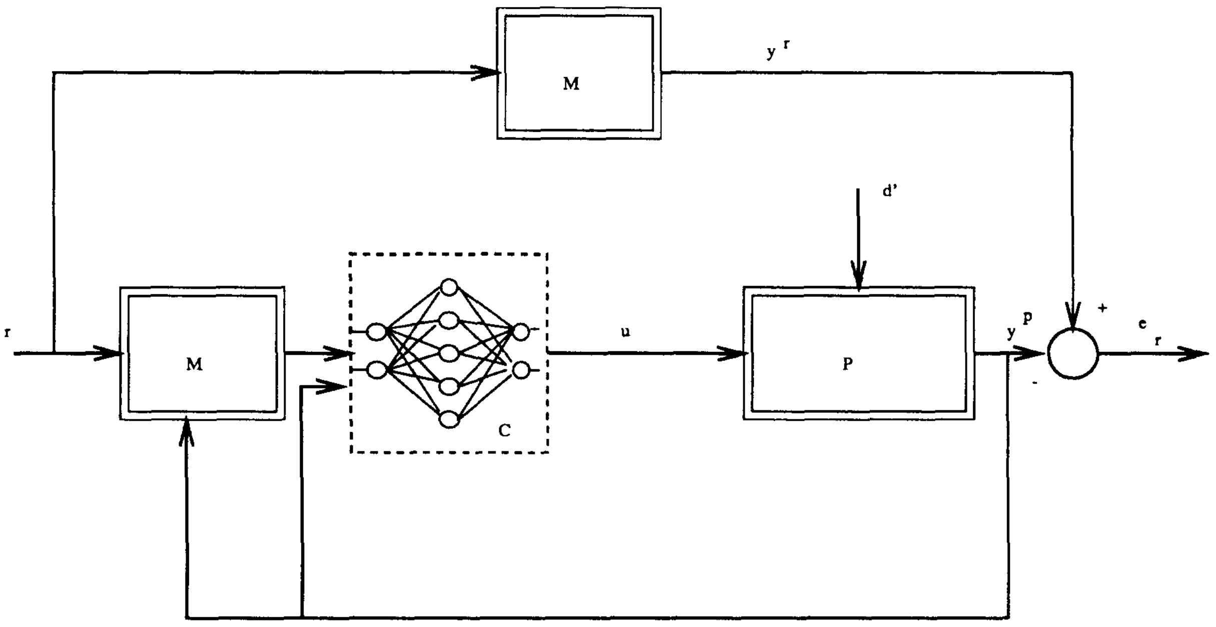

FIG. 6. Model reference structure.

readily to nonlinear systems control (Economou *et al.* (1986)).

In internal model control a system model is placed in parallel with the real system. The difference between the system and model outputs is used for feedback purposes. This feedback signal is then processed by a controller subsystem in the forward path; the properties of IMC dictate that this part of the controller should be related to the system inverse (the neural network realization of IMC is illustrated in Fig. 7).

Given network models for the system forward and inverse dynamics the realization of IMC using neural networks is straightforward (Hunt and Sbarbaro (1991)); the system model *M* and the controller *C* (the inverse model) are realized using the neural network models as shown in Fig. 7. The subsystem *F* is usually a linear filter which can be designed to introduce desirable robustness and tracking response to the closed-loop system.

It should be noted that the implementation structure of IMC is limited to open-loop stable systems. However, the technique has been widely applied in process control. From the control theoretic viewpoint there is strong support for the IMC approach. Examples of the use of neural networks for

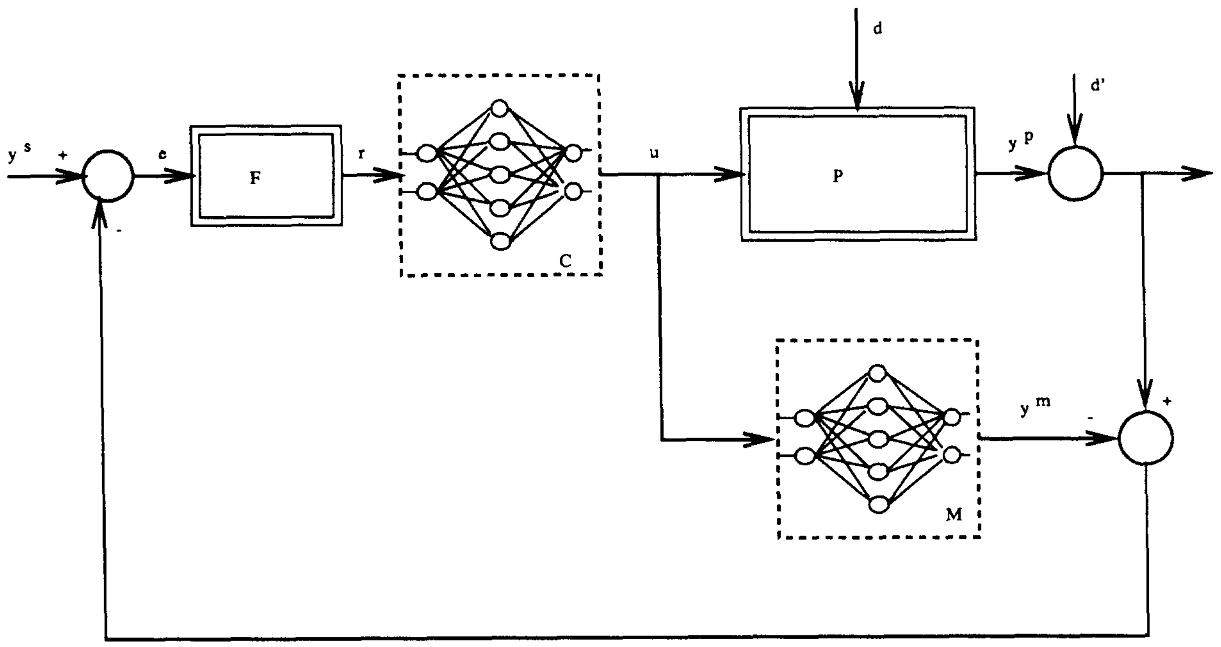

FIG. 7. Structure for internal model control.

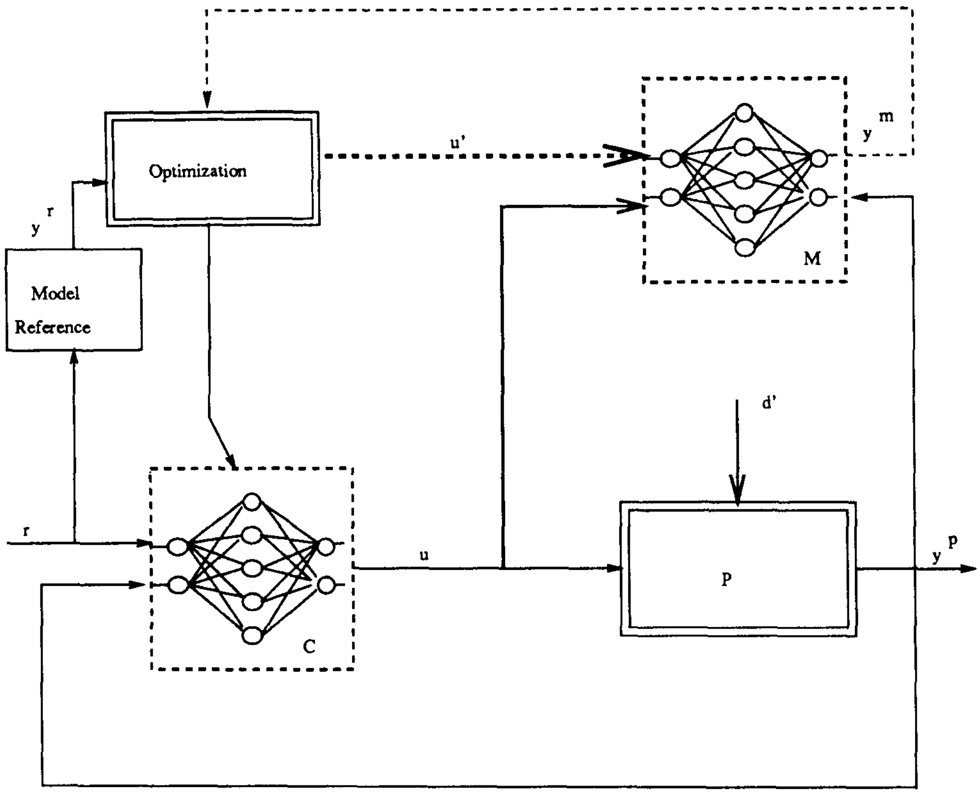

FIG. 8. Structure for predictive control.

nonlinear IMC can be found in Hunt and Sbarbaro (1991).

5.3.5. *Predictive control.* In the realm of optimal and predictive control methods the receding horizon technique has been introduced as a natural, computationally feasible feedback law. It has been proven that the method has desirable stability properties for nonlinear systems (Keerthi and Gilbert (1986); Mayne and Michalska (1990)).

In this approach a neural network model provides prediction of the future plant response over the specified horizon (see Fig. 8). The predictions supplied by the network are passed to a numerical optimization routine which attempts to minimize a specified performance criterion in the calculation of a suitable control signal.

The control signal u' is chosen to minimize the quadratic performance criterion

$$J = \sum_{j=N_1}^{N_2} (y^r(t+j) - y^m(t+j))^2 + \sum_{j=1}^{N_2} \lambda_j(u'(t+j-1) - u'(t+j-2))^2, \quad (78)$$

subject to the constraint of the dynamical model. Here, the constants N_1 and N_2 define the horizons over which the tracking error and control increments are considered. The values of λ are the control weights.

One further possibility, as illustrated in Fig. 8, is to train a further network to mimic the action of the optimization routine. This controller network C is trained to produce the same control output u, for a given plant output, as the optimization routine (u'). An advantage of this approach is that the outer loop consisting of plant model and optimization routine is no longer needed when training is complete.

6. OTHER APPLICATIONS

In the preceding section we described the use of neural networks in nonlinear control structures which are well established in the control theory literature. In this section we review alternative approaches to nonlinear control from the connectionist standpoint. We also take a more general look at system identification using networks and briefly discuss prediction and filtering.

A resume of the control applications of neural networks can be found in Table 4. Table 5 gives an idea of the actual architectures and their use in control. Below, we give a more detailed explanation of some of the possibilities.

6.1. *System identification*

The identification of plant forward and inverse models was discussed in the previous section. These models were developed with a view to utilizing them in a model based control strategy. In this section we take a more general look at identification using neural networks.

The first step in solving a system identification problem is to select an appropriate class of model structures. The connectionist paradigm provides a set of structures to represent nonlinear systems. In general, however, it is possible that a complete description of the physical system is not feasible, or is

Table 4. Summary of applications

Application	Architecture	Learning	References
Adaptive control using generic index	BP	Reinforcement learning	(Barto (1988); Barto *et al.* (1983); Helferty *et al.* (1989); Anderson (1989); Porcino and Collins (1990); Guhua and Mathur (1990); Chen (1990); Franklin (1989); Lee and Berenji (1989))
Adaptive filtering and prediction	BP	BP learning alg.	(Lapedes and Farber (1987); Casdagli (1989); Chen *et al.* (1990b); Ydstie (1990); Weigand *et al.* (1990))
Adaptive nonlinear control	BP	BP learning alg.	(Goldberg and Pearlmutter (1988); Yeung and Beckey (1989); Bassi and Beckey (1989); Josin (1988); Sekiguchi *et al.* (1989); Chen and Pao (1989); Psaltis *et al.* (1988); Guez and Selinsky (1990); Josin (1990); Grant and Zhang (1989); Sanner and Akin (1990); Chen (1989); Karsai *et al.* (1989); Troudet and Merril (1989); Ciliz and Isik (1989); Bhat and McAvoy (1990); Ydstie (1990); Ungar *et al.* (1990); Savic and Tan (1989); Bhat *et al.* (1990))
	CMAC	Delta rule	(Atkenson and Reinkensmeyer (1989); Ersü and Tolle (1984); Kraft and Campagna (1990); Miller *et al.* (1987))
	Kohonen and linear system	Kohonen alg. and delta rule	(Graf and Lalonde (1988); Martinez *et al.* (1988); Ritter and Schulten (1986, 1988))
	CPN	Kohonen alg. and delta rule	(Cheok and Huang (1989))
Optimal control	Kohonen and linear system	Kohonen alg. and delta rule	(Fu (1970))
Nolinear modelling	BP	BP learning alg.	(Bhat and McAvoy (1990); Bhat *et al.* (1990))
Adaptive linear control	Hopfield		(Chi *et al.* (1990))
	ART II		(Kumar and Guez (1990))
	Hopfield		(Żak (1990))
Gain scheduling	Hopfield		(Guez *et al.* (1988))

not practical, and a compromise has to be made between the model complexity and its adequacy for a given application (Goodwin and Sin (1984)). The questions of overparametrization and identifiability are discussed in Section 7.1.2.

The connectionist approach can be located in a corner of a triangle (Bassi (1990)) where the different approaches to the modelling of nonlinear systems are shown (Fig. 9). Each architecture oriented to nonlinear representation can be located in some part of this triangle. It should be noted, however, that the connectionist approach is not confined to the corner of the triangle shown; in fact, neural networks can be used as the implementation mechanism for the other approaches.

The lookup table (tabular) approach is based on a position of a table that contains the value of the function. To address this position an algorithm to detect features in the inputs signals can be used (i.e. LVQ). Some authors (Ritter and Schulten (1986, 1988)) have refined this mechanism allowing each entry be a linear model. A more general approach is the use of overlapping regions instead of disjoint regions. This approach gives the possibility of generalization around a neighbourhood. CMAC and CPN provide an alternative to this approach. Other methods that have been used widely include Potential Functions (Aizerman *et al.* (1964)) and, related with this, Gaussian networks. In the Gaussian case the functions depend upon the distance from a determined point in the space. In this way an interpolation capability is achieved.

In the algebraic representation, on the other hand, there is a set of basic nonlinear functions that forms a base for the model. In some cases it is necessary to use powers of the inputs and their combinations (Billings (1985); Giles and Maxwell (1987)). In other cases it is possible to include some knowledge of the

Table 5. Cross-reference table

	Architectures				
Applications	BP	CMAC	Kohonen	ART II	Hopfield
Adaptive control using generic index	X				
Adaptive filtering and prediction	X				
Adaptive nonlinear control	X	X	X		
Optimal control			X		
Nonlinear modelling	X				
Estimation for time varying systems					X
Adaptive linear control				X	X
Gain scheduling	X				

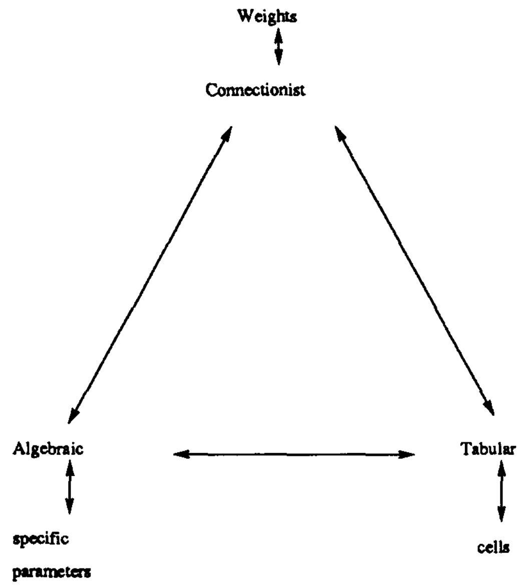

FIG. 9. Connectionist approach in the representation framework.

plant (Kawato *et al.* (1988); Johanson (1990)). In all these cases the nonlinear system is transformed to a linear in the parameters model.

Finally, looking at the third corner of our triangle, the connectionist approach has the advantage that it uses a type of generic nonlinearity and allows all the parameters to be adjusted. In this way it can deal with a very wide range of nonlinearities. Techniques for estimation of the network parameters vary according to the particular architecture used. However, the general structure, which is shown in Fig. 4, can be regarded as invariant.

All these architectures are used in temporal processing provided that the input is a temporal window with data. As has been mentioned in Section 4, however, there are some drawbacks with this approach. The extension to temporal learning in control is an open field to be addressed.

6.2. *Optimal decision control*

In this application, the state space is partitioned into regions (feature space) corresponding to various control situations (pattern classes). The realization of the control surface is accomplished through a training procedure. Since the time-optimal surface in general is nonlinear it is necessary to use an architecture capable to approximating a nonlinear surface. One possibility is to first quantize the state space into elementary hypercubes in which the control action is assumed constant. This process can be carried out with a LVQ architecture. It is then necessary to have another network acting as a classifier. If continuous signals are required a standard back-propagation architecture can be used (see Fig. 10).

The switching surface is not known *a priori* but is defined implicitly by a training set of points in state space whose optimal control action is known. During the training process the learning algorithm makes changes in its weights based only on the training pattern vector presently being shown to it, together with the desired control situation. The training pattern vectors are presented to the controller sequentially several times until all the pattern vectors in the training set are being correctly classified, or until the number of classification errors has reached some steady-state value (Fu (1970)).

6.3. *Adaptive linear control*

The connectionist approach can be used in nonlinear control but also as a part of a controller for linear plants.

The Hopfield network may be utilized as a dynamical controller (Żak (1990)) for a linear system. In this case, elements of variable structure theory were utilized to construct the controller, and as a result the proposed controller is characterized by its robustness. It is possible to include adaptation using some standard method (Demircioglu and Gawthrop (1988)). The general structure is shown in Fig. 11.

The Hopfield network can also be used as a part of the adaptation mechanism (Chi *et al.* (1990)). In this case the network is included in the adaptation path and is used to minimize simultaneously square-error rates of all the states. Here, the output of the network represents the parameters of a linear model for the plant. This approach has been shown to be useful for implementation of a least-squares estimation for time-varying and time-invariant systems.

6.4. *Adaptive control using a generic index (reinforcement learning control)*

This kind of network makes use of a low quality feedback signal. While the performance measure for a

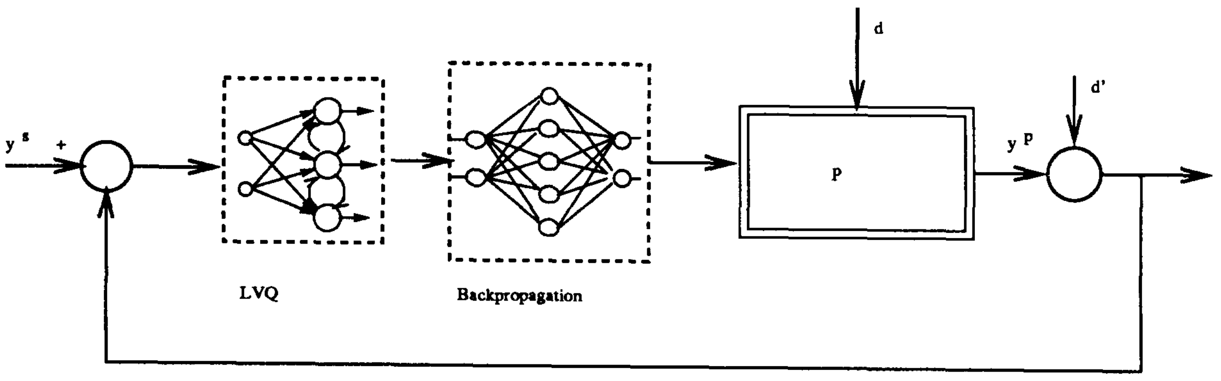

FIG. 10. Structure for optimal control.

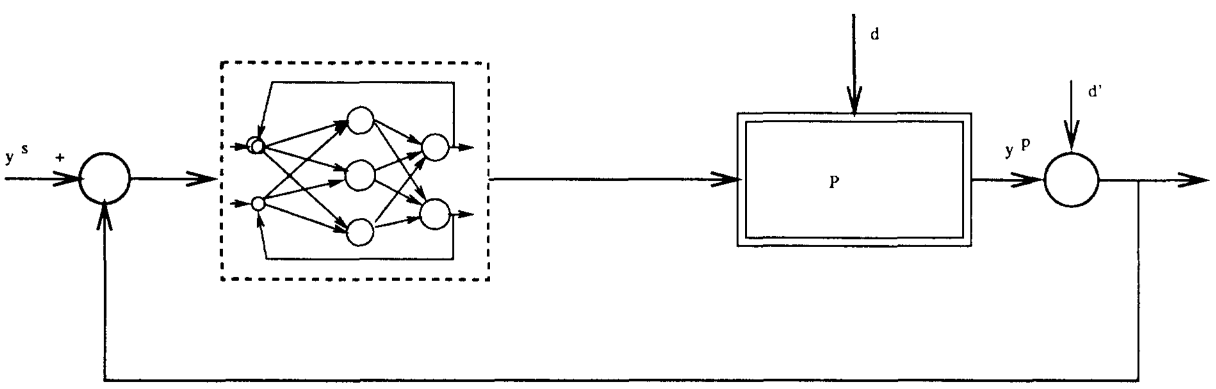

FIG. 11. Structure for feedback networks as controllers.

supervised system is defined in terms of a set of targets by means of a known error, reinforcement learning addresses the problem using any measure whose value can be supplied to the learning algorithm.

Instead of trying to determine target controller outputs from target plant responses, one tries to determine target controller outputs that would lead to an increase in a measure of plant performance (a measure not necessarily defined in terms of target plant responses). The "critic" block is capable of evaluating the plant performance and generates an "evaluation signal" which is used by the reinforcement learning algorithm. This approach is appropriate when there is a genuine lack of knowledge required for applying more specialized learning methods. An example is the pole-cart problem where in a reinforcement scheme the evaluation is zero throughout a trial and becomes -1 when a failure occurs. In the more knowledge intensive controller a reference trajectory is necessary to control the plant.

A comprehensive discussion of reinforcement learning control and a list of references can be found in Barto (1990).

6.5. *Gain scheduling*

This is an old control engineering technique which uses process variable related with its dynamics to compensate the effect of working in different operating regions. Here, the neural network is used as an associative memory that relates the parameters of the controller with the state of the plant. The Hopfield network has been used in this case (Guez *et al.* (1988)) where its output corresponds to the parameters of the controller. Thus, the stable-state topology can be designed so that the local minima correspond to optimal control laws for the parameters of the controller. Other connectionist architectures, such as those based on receptive fields (for example, CMAC—see Section 2.5) can be used to obtain the same results.

6.6. *Filtering and prediction*

Filtering is concerned with the extraction of signals from noise. In this case the connectionist approach is used to produce a nonlinear filter such that the effect of the noise is cancelled. In the past the general least-mean-square approach has been extended on the basis of a truncated Wiener–Volterra series representation. This series is a general representation for a large class of nonlinear systems, and has the advantage that the output is linear with respect to the filter weights. This approach suffers from the drawback that the computation and implementation cost increases sharply with the nonlinearity. Thus, it has been used only in weak-nonlinear cases (Lin and Unbehauen (1990)). In this case the connectionist approach has the advantage of saving computation and implementation costs, especially in strongly-nonlinear control. In the special case that the model of the noise is a chaotic system in a linear process it is possible to find an expression for the equivalent linear minimum variance filter (Ydstie (1990)). For a presentation of some non-ANN approaches to the control of chaotic systems the reader is directed to Ditto *et al.* (1990) and Ott *et al.* (1990).

Prediction is defined as the problem of extrapolating a given time series into the future. As for filtering, if the underlying model of the time series is known it is possible in principle to design an optimal predictor for future values. The design is based in this case on connectionist models. Prediction in the context of receding horizon control was discussed in the preceding section.

As mentioned in Lapedes and Farber (1987) a competing method for processing nonlinear signals is the algebraic approach using polynomials (see also Farmer and Sidorowich (1987)). This has the advantage that polynomial nonlinearities can be modelled exactly. On the other hand, non-polynomial nonlinearities must also be modelled by polynomials. This can lead to a rapid oscillation of the polynomials, and as the system size increases there is an explosion in the number of polynomials. The reason that connectionist networks seem to work well can be related to the fact that the network has the capability to perform a kind of generalized mode decomposition of underlying maps. That is, the parameters of the basis function are adjusted to change completely their input/output relations.

A discussion of filtering and prediction using neural networks can be found in the paper by Widrow and Winter (1988).

7. OPEN RESEARCH TOPICS

The field of neural networks in control is proceeding, as are most other emergent scientific disciplines, through a mix of empirical and theoretical studies. Not surprisingly, thus far more effort has been concentrated on the former aspect. On the theoretical side progress has been limited; a large number of crucial and highly challenging questions remain open. In this section we make an initial assessment of the major research topics.

Much insight can be gained from adaptive control theory (Åström and Wittenmark (1989)). Twenty years ago this area was characterized by a large body of experimental work which showed great promise. This initial flurry has been followed by a consolidation period where some of the major theoretical questions were formulated and solved; the theory of adaptive systems is now relatively mature. A strong interplay between theory and practice has also been evident in the development of adaptive control.

Adaptive control theory is generally based on the assumption of linear time-invariant plant. Even though the adaptation process results in the overall system being nonlinear the cornerstone of this work lies in linear systems theory (Narenda and Annaswamy (1989)). Many results additionally require a significant level of prior knowledge about the plant.

It is evident, on the other hand, that the primary focus of neural networks will be on nonlinear control problems. The strong analogy between traditional adaptive control and neural network based control indicates that this new area will develop along similar lines taken by adaptive control theory and that similar theoretical difficulties will emerge. Thus, the established results and techniques of linear adaptive control may well provide insight for the case where system components are nonlinear. It is also reasonable to conjecture that the problems encountered will be more complex due to the nonlinearity of the subsystems involved. In order to have a well-developed theory of control using neural networks many fundamental questions will have to be addressed.

7.1. *Systems theory*

A rigorous characterization of theoretical properties such as stability, controllability and observability should be developed.

7.1.1. *Stability.* The crucial question of stability of adaptive systems has generated a large literature. Lyapunov methods and hyperstability theory have been applied to prove the stability of adaptive systems where the underlying plant is assumed linear and time-invariant. Linearization techniques also play an important role.

For adaptive systems composed of nonlinear plants and neural networks new concepts and techniques for stability theory will need to be explored. The results obtained in the context of dynamic networks are far from satisfactory (see Sections 2.4.1 and 2.4.2). Stability issues should also be elaborated for control-orientated neural network structures.

7.1.2. *Persistent excitation and convergence.* The importance of the signals used to train a network has already been emphasised. The training data must sufficiently cover the range of operation signals. This will help to ensure that the network will operate in the desired way when presented with inputs not included in the training set. The issue of how well the model output approximates that of the plant after training is known as generalization. A related concept in the adaptive control and system identification literature is persistent excitation. Having persistently exciting training data ensures convergence of the model to the "true" plant.

A related concept is that of system identifiability: can the system under study be adequately represented within the chosen model structure? For example, it is well known in adaptive control that using an overparametrized model can lead to convergence problems. This is indicative of a trade-off between identifiability and generalization: using more parameters to get a better approximation makes it more difficult to actually estimate (from a given training set) the values of parameters that give good generalization.

A rigorous theory covering these issues needs to be developed in the context of neural network models.

7.1.3. *Robustness.* As already mentioned, we believe that results from the field of adaptive control have a role to play in the analysis and design of Neural Networks. Perhaps the most promising category of such results relates to robustness: that is, the study of when properties such as stability and convergence are retained in the presence of system dynamics which cannot be completely modelled by the adaptive controller (Rohrs *et al.* (1985); Kosut and Friedlander (1985); Kosut and Johnson (1984); Anderson *et al.* (1986); Gawthrop (1987); Cluett *et al.* (1988); Sastry and Bodson (1989); Åström (1989); Gawthrop (1990)).

7.2. *Network theory*

In a previous section a body of existence theorems relating to the approximation capabilities of neural networks was described. What is now needed is the development of constructive procedures for design and development of neural networks. The ultimate aim is to achieve a systematic engineering procedure which will guide the selection of a network architecture for a specific problem.

7.2.1. *Suitable applications.* It is desirable to develop a characterization of the kinds of real problems for which neural networks are suitable, and for which they can be expected to improve on the performance of "traditional" control techniques in a meaningful way. Neural networks will be aimed primarily at nonlinear systems. It is important to classify the nonlinear systems for which current nonlinear control techniques have been developed, and hence to identify those classes of nonlinear systems at which neural networks should be aimed.

7.2.2. *Network architecture.* A relationship needs to be established between the system under study (i.e. the nonlinear function to be approximated) and the structure of a suitable network. By structure here we are referring to characteristics such as number of layers, number of nodes, and type of nonlinear activation function. A link also needs to be established between these network parameters and the required approximation accuracy.

The implications of using finite numbers of layers/nodes for function approximation also need examined. This raises questions of robustness of the underlying control strategy to modelling error.

7.2.3. *Structural learning.* Structural learning algorithms aim to empirically determine the network structure in terms of numbers of nodes and layers. Such techniques are complementary to the theoretical (or analytical) approach described above.

7.2.4. *Dynamic systems.* In preceding sections we presented one possibility for using static networks to represent dynamic systems. The use of dynamic recurrent networks to represent dynamic systems will we believe become increasingly important. This raises questions of approximation theory and learning algorithms for dynamical networks.

8. CONCLUSIONS

This paper has provided a survey of artificial neural networks in the realm of modelling, identification and control of nonlinear systems. The basic ideas and techniques of artificial neural networks have been presented in language and notation familiar to control engineers. Applications of a variety of neural network architectures in control were surveyed. We explored the links between the fields of control science and neural networks in a unified presentation and identified key areas for future research.

This survey is aimed at researchers currently working in the area of control systems, by presenting the basic ideas and techniques of artificial neural networks in language and notation familiar to control engineers we hope to have made these techniques more readily accessible to control engineers. Control systems insights can be applied to artificial neural network techniques.

Acknowledgements—Kenneth Hunt currently holds a Royal Society of Edinburgh Personal Research Fellowship. Daniel Sbarbaro is supported by a Chilean government research scholarship. Rafał Żbikowski is supported under SERC research grant GR/G/53569, *Neural Networks for Modelling and Control of Nonlinear Dynamic Systems.* He previously worked on this paper while on leave from Warsaw University of Technology; support for his visit was furnished by the TEMPUS programme of the European Community (grant number IMG-PLS-0630-90). Peter Gawthrop is Wylie Professor of Mechanical Engineering.

The authors would finally like to express their gratitude to the reviewers of this paper who put considerable effort into their task and made many deep and constructive comments.

REFERENCES

Aarts, E. and J. Korst (1989). *Simulated Annealing and Boltzmann Machines.* Wiley, New York.

Åström, K. J. (1989). Application of the robust and adaptive pole placement design technique *Int. J. of Adaptive Control and Signal Processing,* **3,** 167–189.

Åström, K. J. and B. Wittenmark (1989). *Adaptive Control.* Addison-Wesley, New York.

Ackley, D. H., G. H. Hinton and T. J. Sejnowski (1985). A learning algorithm for Boltzmann machines. *Cognitive Science,* **9,** 147–169.

Aizerman, M. A., E. M. Braverman and L. I. Rozonoer (1964). Theoretical foundation of the potential function method in pattern recognition learning. *Automatika i Telemekhanika,* **25,** 1917–1936.

Albert, A. E. and L. A. Gardner (1967). *Stochastic Approximation and Nonlinear Regression.* MIT Press, Cambridge, MA.

Albus, J. S. (1975a). Data storage in the cerebellar model articulation controller (CMAC). *Trans. of the ASME, J. of Dynamic Systems, Measurement, and Control,* **97,** 228–233.

Albus, J. S. (1975b). A new approach to manipulator control: The cerebellar model controller (CMAC). *Trans. of the ASME, J. of Dynamic Systems, Measurement, and Control,* **97,** 220–227.

Almeida, L. B. (1988). Backpropagation in perceptrons with feedback. In R. Eckmiller and Ch. v. d. Malsburg (Eds), *Neural Computers.* Springer-Verlag, Berlin.

Anderson, B. D. O., R. R. Bitmead, C. R. Johnson, P. V. Kokotovic, R. L. Kosut, I. M. Y. Mareels, L. Praly and B. D. Reidle (1986). *Stability of Adapative Systems: Passivity and Averaging Analysis.* MIT Press, Cambridge, MA.

Anderson, C. W. (1989). Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine,* **9,** 31–37.

Anderson, J. A. and E. Rosenfeld. (1988). *Neurocomputing: Foundations of Research.* MIT Press, Cambridge, MA.

Atkenson, C. G. and D. J. Reinkensmeyer. (1989). Using associative content-addressable memories to control robots. In *Proc. of* 1989 *IEEE Int. Conf. on Robotics and Automation,* Scottsdale, Arizona, pp. 1859–1864.

Ballard, D. H. (1988). Cortical connections and parallel processing: Structure and function. In M. Arbib and Hamson (Eds), *Vision, Brain, and Cooperative Computation,* pp. 563–621. MIT Press, Cambridge, MA.

Barron, A. R. (1991). Approximation and estimation bounds for artificial neural networks. In *Proc. 4th Annual Workshop on Computational Learning Theory,* University of California Santa Cruz, U.S.A., pp. 243–249.

Barto, A. G. (1988). An approach to learning control surface by connectionist systems. In M. Arbib and Hanson (Eds), *Vision, Brain and Cooperative Computation,* pp. 665–701. MIT Press, Cambridge, MA.

Barto, A. G. (1990). *Neural Networks for Control,* Chapter 1, pp. 5–58. MIT Press, Cambridge, MA.

Barto, A. G., R. S. Sutton and Ch. W. Anderson (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. on System, Man, and Cybernetics,* **13,** 834–846.

Bassi, D. F. (1990). Connectionist dynamic control of robotic manipulators. D. Phil. thesis, University of Southern California.

Bassi, D. F. and G. A. Beckey (1989). Decomposition of neural network model of robot dynamics: a feasibility study. *Simulation and AI,* **220,** 8–13.

Bavarian, B. (1988). Introduction to neural networks for intelligent control. *IEEE Control Systems Magazine,* **8,** 3–7.

Bhat, N. V. and T. J. McAvoy (1990). Use of neural nets for dynamical modelling and control of chemical process systems. *Computers Chem. Engng.,* **14,** 573–5583.

Bhat, N. V., P. A. Minderman, T. McAvoy and N. S. Wang (1990). Modeling chemical process systems via neural computation. *IEEE Control Systems Magazine,* **10,** 24–29.

Billings, S. A. (1985). Introduction to nonlinear system analysis and identification. In K. Godfrey and P. Jones (Eds), *Signal Processing for Control.* Springer-Verlag, Berlin.

Broomhead, D. S. and D. Lowe (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems,* **2,** 321–355.

Bruck, J. (1989) Computing with Networks of Threshold Elements. D. Phil. Thesis, Stanford University.

Bullock, D. (1983). How neural networks factor problems of sensory motor control. In *Proc. American Control Conf.*, pp. 2271–2275.

Burkill, J. C. and H. Burkill (1970). *A Second Course in Mathematical Analysis*. Cambridge University Press, Cambridge, U.K.

Carrol, M. S. and B. W. Dickinson (1989). Construction of neural nets using the Radon transform. In *Proc. IJCNN*.

Casdagli, M. (1989). Non-linear system prediction of chaotic time series. *Physica D*, **35**, 335–356.

Chen, Fu-Chuang (1989). Back-propagation neural network for nonlinear self-tuning adaptive control. In *IEEE Int. Symposium on Intelligent Control* 1989, pp. 274–279.

Chen, S., S. A. Billings, C. F. Cowan and P. M. Grant (1990a). Practical identification of NARMAX models using radial basis functions. *Int. J. Control*, **52**, 1327–1350.

Chen, S., S. A. Billings and P. M. Grant (1990b). Non-linear system identification using neural networks. *Int. J. Control*, **51**, 1191–1214.

Chen, S., C. F. N. Cowan, S. A. Billings and P. M. Grant (1990c). Parallel recursive prediction error algorithm for training layered neural networks. *Int. J. Control*, **51**, 1215–1228.

Chen, V. C. and Y. H. Pao (1989). Learning control with neural networks. In *Proc. of* 1989 *IEEE Int. Conf. on Robotics and Automation*, Scottsdale, AZ. pp. 1448–1453.

Chen, V. C. (1990). Problem-solving by using reinforcement learning neural nets. In *IEEE Int. Joint Conf. on Neural Networks, IJCNN'*90, pp. 583–586.

Cheok, K. C. and N. J. Huang (1989). Lyapunov stability analysis for self-learning neural model with application to semiactive suspension control system. In *IEEE Int. Symposium on Intelligent Control* 1989, pp. 329–331.

Chester, D. (1990). Why two hidden layers are better than one. In *IEEE Int. Joint Conf. on Neural Networks, IJCNN'*90, pp. 265–268.

Chi, S. R., R. Shoureshi and M. Tenorio (1990). Neural networks for system identification. *IEEE Control Systems Magazine*, **10**, 31–34.

Chua, L. O. and L. Yang (1988a). Cellular neural networks: Applications. *IEEE Trans. on Circuits and Systems*, **35**, 1273–1290.

Chua, L. O. and L. Yang (1988b). Cellular neural networks: Theory. *IEEE Trans. on Circuits and Systems*, **35**, 1257–1272.

Ciliz, M. K. and C. Isik (1989). Time optimal control of mobile robot motion using neural nets. In *IEEE Int. Symposium on Intelligent Control* 1989, pp. 368–373.

Cluett, W. R., S. L. Shah and D. G. Fisher (1988). Robustness analysis of discrete-time adaptive control systems using input–output stability theory: A tutorial. *IEE Proc.*, **135**, 133–141.

Cohen, M. A. and S. Grossberg (1983). Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Trans. on System, Man, and Cybernetics*, **13**, 815–826.

Cybenko, G. (1988). Continuous valued networks with two hidden layers are sufficient. Technical Report, Department of Computer Science, Tufts University.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Math. Control Signal Systems*, **2**, 303–314.

Daunicht, W. J. (1990). Defanet—a deterministic approach to function approximation by neural networks. In *IEEE Int. Joint Conf. on Neural Networks, IJCNN'*90, pp. 161–164.

Demircioglu, H. and P. J. Gawthrop (1988). Continuous-time relay self-tuning control. *Int. J. Control*, **47**, 1061–1080.

Ditto, W. L., S. N. Rauseo and M. L. Spano (1990). Experimental control of chaos. *Physical Review Letters*, **65**, 3211.

Dzieliński, A., S. Skoneczny, R. Żbikowski and S. Kukliński (1990). Cellular neural network application to moiré pattern filtering. In *Proc. IEEE CNNA'90 Workshop*, Budapest, Hungary.

Economou, C. G., M. Morari and B. O. Palsson (1986). Internal model control. 5. Extension to nonlinear systems. *Ind. Eng. Chem. Process Des. Dev.*, **25**, 403–411.

Ersü, E. and H. Tolle (1984). A new concept for learning control inspired by brain theory. *Proc. 9th World Congress of IFAC*, **7**, 245–250.

Eykhoff, P. (1974). *System Identification*. Wiley, New York.

Fang, Y. and T. J. Sejnowski (1990). Faster learning for dynamic recurrent backpropagation. *Neural Computation*, **2**, 270–273.

Farmer, D. and J. Sidorowich (1987). Predicting chaotic time series. *Physical Review Letters*, **59**, 845.

Fletcher, R. (1987). *Practical Methods of Optimization*, 2nd ed. Wiley, Chichester.

Franklin, J. A. (1989). Input space representation for refinement learning control. In *IEEE Int. Symposium on Intelligent Control* 1989, pp. 115–122.

Fu, K. S. (1970). Learning control systems—review and outlook. *Trans. IEEE on Aut. Control*, **16**, 210–221.

Funahashi, K. I. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks*, **2**, 183–192.

Garcia, C. E. and M. Morari (1982). Internal model control—1. A unifying review and some new results. *Ind. Eng. Chem. Process Des. Dev.*, **21**, 308–323.

Gawthrop, P. J. (1987). Robust stability of a continuous-time self-tuning controller. *Int. J. of Adaptive Control and Signal Processing*, **1**, 31–48.

Gawthrop, P. J. (1990). Robust stability of multi-loop continuous-time self-tuning controllers. *Int. J. of Adaptive Control and Signal Processing*, **4**, 359–382.

Gawthrop, P. J. and D. G. Sbarbaro (1990). Stochastic approximation and multilayer perceptrons: The gain back-propagation algorithm. *Complex System J.*, **4**, 51–74.

Giles, C. L. and T. Maxwell (1987). Learning, invariance, and generalization in high-order neural networks. *Applied Optics*, **26**, 4972–4978.

Girosi, F. and T. Poggio (1989). Representation properties of networks: Kolmogorov's theorem is irrelevant. *Neural Computation*, **1**, 465–469.

Girosi, F. and T. Poggio (1990). Networks and the best approximation property. *Biological Cybernetics*, **63**, 169–176.

Goldberg, K. and B. A. Pearlmutter (1988). Using a neural network to learn the dynamic of the CMU direct-drive arm II. Report CMU-CS-88-160, Carnegie-Mellon University, 1988.

Goodwin, G. C. and K. S. Sin (1984). *Adaptive Filtering Prediction and Control*. Prentice-Hall, Englewood Cliffs, NJ.

Graf, D. H. and W. R. Lalonde (1988). A neural controller for collision-free movement of general robot manipulators. In *IEEE Int. Joint Conf. on Neural Networks, IJCNN'*88, pp. 77–84.

Grant, E. and B. Zhang (1989). A neural net approach to supervised learning of pole balancing. In *IEEE Int. Symposium on Intelligent Control* 1989, pp. 123–129.

Grossberg, S. (1976). Adaptive pattern classification and universal recording: I. parallel development and coding of neural feature detectors. *Biological Cybernetics*, **23**, 121–134.

Grossberg, S. (1988). *Neural Networks and Natural Intelligence*. The MIT Press, Cambridge, MA.

Guez, A., J. L. Elibert and M. Kam (1988). Neural network architecture for control. *IEEE Control Systems Magazine*, **8**, 22–25.

Guez, A. and J. W. Selinsky (1990). A neurocontroller with guaranteed performance for rigid robots. In *IEEE Int. Joint Conf. on Neural Networks, IJCNN'*90, pp. 347–350.

Guhua, A. and A. Mathur (1990). Setpoint control based on reinforcement learning. In *IEEE Int. Joint Conf. on Neural Networks, IJCNN'*90, pp. 511–514.

Haber, R. and Unbehauen, H. (1990). Structure identification of nonlinear dynamic systems—A survey on input/output approaches. *Automatica*, **26**, 651–677.

Hartman, E. and J. D. Keeler (1992). Predicting the future with semi-local units. *Neural Computation*. To appear.

Hebb, D. O. (1949). *The Organization of Behaviour*. Wiley, New York.

Hecht-Nielsen, R. (1987a). Counterpropagation networks. *Applied Optics*, **26**, 4979–4984.

Hecht-Nielsen, R. (1987b). Kolmogorov's mapping neural network existence theorem. In *Proc. IJCNN*.

Hecht-Nielsen, R. (1988). Neurocomputer applications. In R. Eckmiller and Ch. v. d. Malsburg (Eds), *Neural Computers*, pp. 445–453. Springer-Verlag, Berlin.

Helferty, J. J., J. B. Collins and M. Kam (1989). A neural network learning strategy for the control of a one-legged hopping machine. In *Proc. of* 1989 *IEEE Int. Conf. on Robotics and Automation*, Scottsdale, AZ, pp. 1604–1609.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proc. of the National Academy of Sciences*, **79**, 2554–2558.

Hopfield, J. J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. of the National Academy of Sciences*, **81**, 3088–3092.

Hornik, K., M. Stinchcombe and H. White (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, **2**, 359–366.

Hunt, K. J. and D. Sbarbaro (1991). Neural networks for non-linear internal model control. *Proc. IEE Pt. D*, **138**, 431–438.

IEEE (1988). Special issue on neural networks. *IEEE Control Systems Magazine*, **8**.

IEEE (1989). Special issue on neural networks. *IEEE Control Systems Magazine*, **9**.

IEEE (1990). Special issue on neural networks. *IEEE Control Systems Magazine*, **10**.

Johanson, R. (1990). Real-time identification of multilinear systems. In *Preprints of the* 11*th IFAC World Congress*, Tallin, Estonia, Vol. 3, pp. 119–123.

Johnson, J. L. (1991). Globally stable saturable learning laws. *Neural Networks*, **4**, 47–51.

Jordan, M. I. and D. E. Rumelhart (1991). Forward models: supervised learning with a distal teacher, Occasional Paper #40, Center for Cognitive Science, Massachusetts Institute of Technology (submitted to "Cognitive Science"); 49 pages.

Josin, G. (1988). Neural-space generalization of a topological transformation. *Biological Cybernetics*, **59**, 283–290.

Josin, G. M. (1990). Development of a neural network autopilot model for high performance aircraft. In *IEEE Int. Joint Conf. on Neural Networks, IJCNN'*90, pp. 547–550.

Karsai, G., K. Anderson, G. Cook and K. Ramaswamy (1989). Dynamic modelling and control of nonlinear process using neural network techniques. In *IEEE Int. Symposium on Intelligent Control* 1989, pp. 280–286.

Kawato, M., Y. Uno, M. Isobe and R. Suzuki (1988). Hierarchical neural network model for voluntary movement with application to robotics. *IEEE Control Systems Magazine*, **8**, 8–17.

Keerthi, S. S. and E. G. Gilbert (1986). Moving-horizon approximations for a general class of optimal nonlinear infinite-horizon discrete-time systems. In *Proc.* 20*th Annual Conf. Information Science and Systems*, Princeton University, pp. 301–306.

Kelly, D. G. (1990). Stability in contractive nonlinear neural networks. *IEEE Trans. on Biomedical Engineering*, **37**, 231–242.

Kohonen, T. (1987). *Self-Organization and Associative Memory*. Springer-Verlag, Berlin.

Kollias, S. and D. Anastassiou (1989). An adaptive least squares for efficient training of artificial neural networks. *IEEE Trans. on Circuits and Systems*, **36**, 1092–1101.

Kolmogorov, A. N. (1957). On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition. *Dokl. Akad. Nauk SSSR*, **114**, 953–956.

Kosikov, V. S. and A. P. Kurdyukov (1987). Design of a nonsearching self-adjusting system for nonlinear plant. *Automatika i Telemekhanika*, **4**, 58–65.

Kosut, R. L. and B. Friedlander (1985). Robust adaptive control: conditions for global stability. *IEEE Trans. on Aut. Control*. **AC-30**, 610–624.

Kosut, R. L. and C. R. Johnson (1984). An input-output view of robustness in adaptive control. *Automatica*, **20**, 569–582.

Kraft, L. G. and D. P. Campagna (1990). A comparison between CMAC neural network control and two traditional adaptive control systems. *IEEE Control Systems Magazine*, **10**, 36–43.

Kumar, S. and A. Guez (1990). Adaptive pole placement for neurocontrol. In *IEEE Int. Joint Conf. on Neural Networks, IJCNN'*90, pp. 397–400.

Kurkova, V. (1991). Kolmogorov's theorem is relevant. *Neural Computation*, **3**, 617–622.

Lane, S. H., D. A. Handelman and J. J. Gelfand (1991). Higher order CMAC neural networks—theory and practice. In *Proc. American Control Conf.*, Boston, MA. pp. 1579–1585.

Lapedes, A. and R. Farber (1987). Non-linear signal processing using neural networks: prediction and system modelling. Report LA-UR-87-2662, Los Alamos National Laboratory.

Lee, Chuen-Chien and H. R. Berenji (1989). An intelligent controller based on approximate reasoning and reinforcement learning. In *IEEE Int. Symposium on Intelligent Control* 1989, pp. 200–205.

Li, J. H., A. N. Michel and W. Porod (1988). Qualitative analysis and synthesis of a class of neural networks. *IEEE Trans. on Circuits and Systems*, **35**, 976–986.

Lin, J. and R. Unbehauen (1990). Adaptive nonlinear digital filter with canonical piecewise-linear structure. *IEEE Trans. on Circuits and Systems*, **37**, 347–353.

Lippmann, R. P. (1987). An introduction to computing with neural nets. *IEEE ASSP Magazine*, **4**, 4–22.

Ljung, L. (1987). *System Identification—Theory for the User*. Prentice Hall, Englewood Cliffs, NJ.

Ljung, L. and T. Söderström (1983). *Theory and Practice of Recursive Identification*. MIT Press, London.

Lorentz, G. G. (1976). *Mathematical Developments Arising from Hilbert's Problems*, Vol. 2, pp. 419–430. American Mathematical Society.

Martinez, T. M., H. Ritter and K. J. Schulten (1988). Three-dimensional neural net for learning visuomotor-coordination of a robot arm. In *IEEE Int. Joint Conf. on Neural Networks, IJCNN'*88, pp. 351–356.

Mayne, D. Q. and H. Michalska (1990). Receding horizon control of nonlinear systems. *Trans. IEEE on Aut. Control*, **35**, 814–824.

McCulloch, W. S. and W. Pitts (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, **9**, 127–147.

Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller (1953). Equation of state calculations by fast computing machines. *J. Chemical Physics*, **21**, 1087–1092.

Michel, A. N., J. A. Farrell and W. Porod (1989). Qualitative analysis of neural networks. *IEEE Trans. on Circuits and Systems*, **36**, 229–243.

Miller, W. T., F. H. Glanz and L. G. Kraft (1987). Application of a general learning algorithm to the control of robotic manipulators. *The Int. J. of Robotic Research*, **6**, 84–98.

Miller, W. T., R. S. Sutton and P. J. Werbos (1990). *Neural Networks for Control*. MIT Press, Cambridge, MA.

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

Minsky, M. L. and S. A. Papert (1988). *Perceptrons (expanded edition)*. The MIT Press, Cambridge, MA.

Moody, J. and C. Darken (1989). Fast-learning in networks of locally-tuned processing units. *Neural Computation*, **1**, 281–294.

Morari, M. and E. Zafiriou (1989). *Robust Process Control*. Prentice-Hall, Englewood Cliffs, NJ.

Narendra, K. S. (1990). *Neural Networks for Control*, Chapter 5, pp. 115–142. MIT Press. Cambridge, MA.

Narendra, K. S. and A. M. Annaswamy (1989). *Stable Adaptive Systems*. Prentice-Hall, Englewood Cliffs, NJ.

Narendra, K. S. and K. Parthasarathy (1990). Identification and control for dynamic systems using neural networks. *IEEE Trans. on Neural Networks*, **1**, 4–27.

Ott, E., C. Grebori and J. A. Yorke (1990). Controlling chaos. *Physical Review Letters*, **64**, 1196.

Park, J. and S. Lee (1990). Neural computation for collision-free path planning. In *Proc. of IEEE Int. Joint Conf. on Neural Networks, IJCNN'90*, Washington, DC.

Pearlmutter, B. A. (1988). Learning state space trajectories in recurrent neural networks. Report CMU-CS-88-191, Carnegie-Mellon University, 1988.

Pearlmutter, B. A. (1989). Learning state space trajectories in recurrent neural networks. *Neural Computation*, **1**, 263–269.

Pearlmutter, B. A. (1990). Dynamic recurrent neural networks. Technical Report CMU-CS-90-196, Carnegie Mellon University, School of Computer Science.

Pineda, F. J. (1987). Generalization of back-propagation to recurrent neural networks. *Physical Review Letters*, **59**, 2229–2232.

Pineda, F. J. (1989). Recurrent backpropagation and the dynamical approach to adaptive neural computation. *Neural Computation*, **1**, 161–172.

Poggio, T. (1982). *Physical and Biological Processing of Images*, pp. 128–153. Springer-Verlag, Berlin.

Poggio, T. and F. Girosi (1990). Networks for approximation and learning. *Proc. of IEEE*, **78**, 1481–1497.

Porcino, D. P. and J. C. Collins (1990). An application of neural networks to the guidance of free-swimming submersibles. In *IEEE Int. Joint Conf. on Neural Networks, IJCNN'90*, pp. 417–420.

Poteryaiko, I. Y. (1991). On Liapunov function for neural networks models with multi-unit interaction. In T. Kohonen, K. Mäkisara, O. Simula and J. Kangas (Eds), *Artificial Neural Networks*, pp. 21–23. Elsevier, North-Holland.

Psaltis, D., A. Sideris and A. A. Yamamura (1988). A multilayered neural network controller. *IEEE Control Systems Magazine*, **8**, 17–21.

Ritter, H. and K. Schulten (1986). Topology conserving mappings for learning motor tasks. In J. S. Denher (Ed.), *Neural Networks for Computing, Aip Conf. Proc.*, **151**, 376–380.

Ritter, H. and K. Schulten, (1988). Extending Kohonen's self-organizing mapping algorithm to learn ballistic movements. In R. Eckmiller and Ch. v. d. Malsburg (Eds), *Neural Computers*, pp. 393–406. Springer-Verlag, Berlin.

Robinson, A. J. (1989). Dynamic error propagation networks. D.Phil. thesis, Cambridge University, U.K.

Robinson, A. J. and F. Fallside (1987). Static and dynamic error propagation networks with application to speech coding. In D. Z. Anderson (Ed.) *Proc. of Neural Information Processing Systems*. American Institute of Physics, 1987.

Rohrs, C. E., L. S. Valavani, M. Athans and G. Stein (1985). Robustness of continuous-time adaptive control in the presence of unmodeled dynamics. *Trans. IEEE*, **AC-30**, 881–889.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, **65**, 386–408.

Rudin, W. (1976). *Principles of Mathematical Analysis*, 3rd ed. McGraw-Hill, Auckland.

Rumelhart, D. E., G. E. Hinton and R. J. Williams (1986). Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland (Eds), *Parallel Distributed Processing*. MIT Press, Cambridge, MA.

Rumelhart, D. E. and J. L. McClelland (1986). *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, Vol. 1: *Foundations*. MIT Press, Cambridge, MA.

Sanner, R. M. and D. L. Akin (1990). Neuromorphic pitch attitude regulation of an underwater telerobot. *IEEE Control Systems Magazine*, **10**, 62–67.

Sastry, S. S. and Bodson (1989). *Adaptive Control: Stability, Convergence, and Robustness*. Prentice-Hall, Englewood Cliffs, NJ.

Sato, M (1990). Real time learning algorithm for recurrent analog neural networks. *Biological Cybernetics*, **62**, 237–241.

Savic, M. and Seow-Hwee Tan (1989). A new class of neural networks suitable for intelligent control. In *IEEE Int. Symposium on Intelligent Control* 1989, pp. 418–423.

Sbarbaro, D. G. and P. J. Gawthrop (1991). Self-organization and adaptation in Gaussian networks. In *9th IFAC/IFORS Symposium on Identification and System Parameter Estimation*, Budapest, Hungary, pp. 454–459.

Scalero, R. S. and N. Tepedelenlioglu (1990). A fast algorithm for neural networks. In *IEEE Int. Joint Conf. on Neural Networks, IJCNN'90*, pp. 77–84.

Sekiguchi, M., S. Nagata and K. Asakawa (1989). Behaviour control for mobile robot by multi-hierachical neural network. In *Proc. of* 1989 *IEEE Int. Conf. on Robotics and Automation*, Scottsdale, AZ, pp. 1578–1583.

Shamma, S. (1989). Spatial and temporal processing in central auditory networks. In Ch. Koch and I. Segev (Eds), *Methods in Neural Modeling*, pp. 247–289. The MIT Press, Cambridge, MA.

Söderström, T. and P. Stoica (1989). *System Identification*. Prentice-Hall, Hemel Hempstead, U.K.

Sudharsanan, S. I. and M. K. Sundareshan (1990). Equilibrium uniqueness and global exponential stability of a neural network for optimization applications. In *Proc. IEEE Int. Joint Conf. on Neural Networks, IJCNN'90*, Washington, DC.

Tan, S., L. Vandenberghe and J. Vandewalle (1990). Remarks on the stability of asymmetric dynamical neural networks. In *Proc. IEEE Int. Joint Conf. on Neural Networks, IJCNN'90*, San Diego, CA.

Troudet, T. and W. Merril (1989). Neoromorphic learning continuous valued mappings in presence of noise. In *IEEE Int. Symposium on Intelligent Control* 1989, pp. 312–319.

Tsypkin, Ya. Z (1971). *Adaptation and Learning in Automatic Systems*. Academic Press, New York.

Ungar, L. H., B. A. Powell and S. N. Kamens (1990). Adaptive networks for fault diagnosis and process control. *Computers Chem. Engng.*, **14**, 561–572.

Weigand, A. S., B. A. Huberman and D. E. Rumelhart (1990). Predicting the future: a connectionist approach. *Int. J. Neural Systems*, **3**, 193.

Werbos, P. J. (1974). Beyond regression: new tools for prediction and analysis in the behavior sciences. Ph.D. Thesis, Harvard University, Committee on Applied Mathematics.

Werbos, P. J. (1989). Maximizing long-term gas industry profits in two minutes in lotus using neural network methods. *IEEE Trans. on Systems, Man, and Cybernetics*, **19**, 315–333.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it? *Proc. of IEEE*, **78**, 1550–1560.

Widrow, B. (1986). Adaptive inverse control. In *Preprints of the 2nd IFAC Workshop on Adaptive Systems in Control and Signal Processing*, Lund, Sweden, pp. 1–5.

Widrow, B. and M. E. Hoff (1960). Adaptive switching circuits. 1960 *IRE WESCON Convention Record, New York: IRE*, pp. 96–104.

Widrow, B. and M. A. Lehr (1990). 30 years of adaptive neural networks: Perceptron, medaline, and backpropagation. *Proc. of the IEEE*, **78**, 1415–1441.

Widrow, B. and S. D. Stearns (1985). *Adaptive Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ.

Widrow, B. and R. Winter (1988). Neural nets for adaptive filtering and adaptive pattern recognition. *IEEE Computer*, **21**, 25–39.

Wiener, N. (1948). *Cybernetics: or Control and Communication in the Animal and the Machine*. MIT Press, Cambridge, MA.

Williams, R. J. and D. Zipser (1989a). Experimental analysis of the real-time recurrent learning algorithm. *Connection Science*, **1**, 87–111.

Williams, R. J. and D. Zipser (1989b). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, **1**, 270–280.

Williams, R. J. and D. Zipser (1990). Gradient-based learning algorithms for recurrent connectionist networks. Technical Report NU-CCS-90-9, Northeastern University, Boston, College of Computer Science.

Willis, M. J., C. Di Massimo, G. A. Montague, M. T. Tham and A. J. Morris (1991). On artificial neural networks in process engineering. *Proc. IEE Pt. D*, **138**, 256–266.

Wolfram, S. (1986). *Theory and Applications of Cellular Automata*. Singapore World Scientific, Singapore.

International Workshop. Cellular neural networks and their applications. *Proc. CNNA'90*.

Ydstie, B. E. (1990). Forecasting and control using adaptive connectionist networks. *Computers Chem. Engng.*, **14**, 583–599.

Yeung, D. Y. and G. A. Beckey (1989). Using a context-sensitive learning network for robot arm control. In *Proc. of* 1989 *IEEE Int. Conf. on Robotics and Automation*, Scottsdale, AZ, pp. 1441–1447.

Zak, S. H. (1990). Robust tracking control of dynamic systems with neural networks. In *IEEE Int. Joint Conf. on Neural Networks, IJCNN'90*, pp. 563–566.

Zhao, X. and J. Mendel (1988). An artificial neural minimum-variance estimator. In *IEEE Int. Joint Conf. on Neural Networks, IJCNN'88*, pp. 499–506.

Zipser, D. (1989). A subgrouping strategy that reduces complexity and speeds up learning in recurrent networks. *Neural Computation*, **1**, 552–558.