Pergamon

Mathl. Comput. Modelling Vol. 23, No. 1/2, pp. 101–117, 1996
Copyright©1996 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0895-7177/96 $15.00 + 0.00

0895-7177(95)00221-9

Neurocontrol: A Literature Survey

S. N. BALAKRISHNAN
Department of Mechanical and Aerospace Engineering and Engineering Mechanics
University of Missouri-Rolla, Rolla, MO 65401, U.S.A.

R. D. WEIL
Advanced Development Center
Electronics & Space Corporation, St. Louis, MO 63136, U.S.A.

Abstract—This paper contains a literature review in Neural Control. The review includes seventy-five annotated citations in experimental and theoretical Neural Control applications. Additionally, another thirty-six citations are included. A brief introduction to general Neural Networks is included. Basic Neurocontrol topologies and training techniques are also reviewed.

Keywords—Neural control, Fuzzy control, Adaptive control, Neural networks.

1. INTRODUCTION

Artificial Neural Network (ANN), also known as connectionist learning and parallel distributed processing, is finding applications in diverse areas: many branches of engineering, health sciences, cognitive sciences, archeology, finance, etc. In these fields, there are two broad categories of problems that have been solved successfully with the ANN methodology. They are the pattern recognition of different data and identification of systems and the control of (known/unknown) dynamic systems. The goal of this paper is to survey the advances in the control of different kinds of systems with ANN or what is known as neurocontrol.

Hecht-Nielsen in Simpson [1, p. 3] gives the following general definition of an ANN system:

> A neural network is a parallel, distributed information processing structure consisting of processing elements (which can possess a local memory and carry out localized information processing operations) interconnected together with unidirectional signal channels called connections. Each processing element has a single output connection which branches ("fans out") into as many collateral connections as desired (each carrying the same signal—the processing element output signal). The processing element output signal can be any mathematical type desired. All of the processing that goes on within each processing element must be completely local; i.e., it must depend upon the current values of the input signal arriving at the processing element via impinging connections and upon values stored in the processing element's local memory.

Simpson [1, p. 4] gives a simpler, less rigorous definition, " ... an ANN is a nonlinear directed graph with edges that is able to store patterns by changing the edge weights and is able to recall patterns from incomplete and unknown inputs."

A biological neuron is the basic building block of the nervous system. Figure 1 shows a simplified view of a biological neuron. The body cell of the neuron is termed the "soma." Connected to the "soma" are multiple "dendrites" and an "axon." These serve as the mechanism of commu-

Typeset by $\mathcal{A}_{\mathcal{M}}\mathcal{S}$-TEX

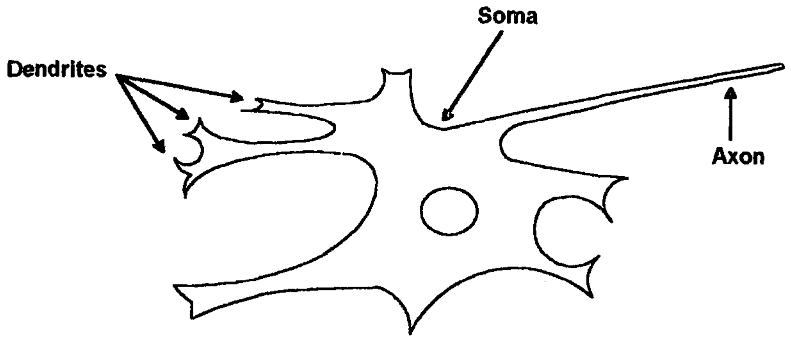


Figure 1. Biological neuron.

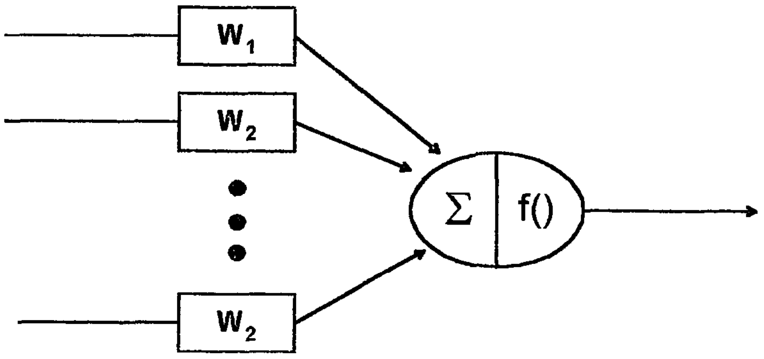


Figure 2. Artificial neuron model.

nication to other neurons. The "dendrites" are spine like connections that receive stimulus from other neurons. Each neuron has a single "axon" that serves to transmit the same stimulus to all the other connected neurons. The connection between a neuron's "axon" and another neuron's "dendrite" is termed a "synapse." Each "synapse" has a level of transmission of the stimulus on the "axon" to the connected "dendrite." Only when a neuron receives sufficient stimulus from other neurons connected at the "dendrites" does the neuron become active and send a stimulus out on its own "axon." Since each neuron accepts a different level of stimulus from a connected neuron based on the transmission level at the "synapse," the network knowledge is equivalent to these transmission levels. Biological neural networks work by a biochemical process that is beyond this survey's scope of interest.

The basic model for an artificial neuron is shown in Figure 2. Lines (or wires) replace the functions of biological "axon" and "dendrites." "Synapses" are replaced by weights (or resistors). The body of the neuron or "soma" is split into two components. The first component is an adder that sums up the weighted outputs of other neurons. The second component is the activation function. This function determines the artificial neuron's activation level based on weighted net input. Typically, this function is an s-shaped function known as a squashing function (e.g., $f(x) \equiv (e^x - 1)/(e^x + 1)$).

The method of connecting and addressing the network neurons and weights defines the network topology. Two general categories of networks exist, recurrent and nonrecurrent. Recurrent networks have cycles in the network connections. These cycles make recurrent networks dynamical systems. Nonrecurrent networks have no cycles and can be viewed as a transformation from n-dimensional Euclidean space to m-dimensional Euclidean space. Figure 3 illustrates a feedforward nonrecurrent network, and a fully recurrent network. Hornik *et al.* have shown that any continuous function may be approximated to an arbitrary degree of accuracy by a feedforward network with one hidden layer, provided there are a sufficient number of hidden units. It is still an open question on how many units are needed, and the methodology for choosing the weights.

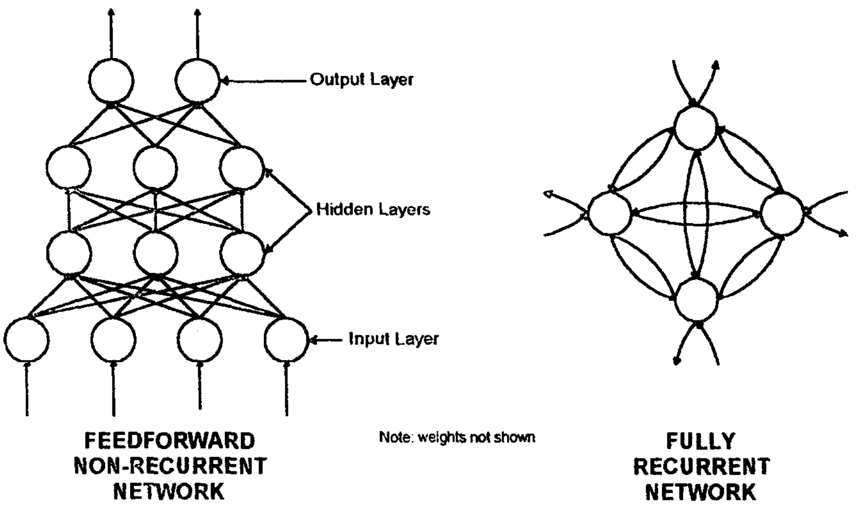


Figure 3. Network topologies.

The early modern research on neural networks can be traced to the work of McCulloch and Pitts [2]. They attempted to translate the events in the nervous system to laws governing information. McCulloch-Pitts networks were designed to perform logical tasks. The mid-modern era started with the work of Rosenblatt [3–5] with his work on Perceptron. Soon after that, Widrow and Hoff [6] introduced the Adaline [7–10]. The difference between Perceptrons and Adaline lies in the training procedure. Rosenblatt was involved with numerical simulation of experiments that he designed the way the human brain is supposed to function. Rosenblatt's neural models are networks of three types of signal processing units: they are the sensory (input) units, the associative units (hidden layer), and the response (output) units. He called his designs 'perceptrons' to underline their *perceptive* abilities. This is quite in contrast to the logic-based artificial intelligence techniques. The underlying philosophy of the perceptron and the subsequently developed neural network architectures are that perception or learned pattern recognition and association abilities are the basis of understanding and skill development. Since the discriminative nature of the human behavior or thinking formed the core of early analysis, several applications of neural networks were in the field of pattern recognition. Such problems used a linear threshold unit. The single layer networks, however, are limited by their inability to separate nonlinear topologies. After Minsky and Papert [11] reported that "some profound reason for the failure to produce an interesting 'learning theorem' for the multilayer machine will be found," the interest in neural network research waned considerably. The extension to solve more complex problems was possible through multilayer networks and the most popular network learning mechanism was formulated with back propagation. Although this work has been credited to Werbos [12] and Rumelhart [13], Dreyfus [14] has shown that Kelly-Bryson gradient formula as recursively derived by him is essentially the same.

The current modern neural network research, which grew with the work of Werbos, received a tremendous boost with the recurrent network introduced by Hopfield [15,16]. In this work, Hopfield presented a model of neural computations which have feedback connections. His hypothesis was that there are emergent computational capabilities at the network level which cannot be found at a single neuron level. With the works of Carpenter and Grossberg [17,18] in adaptive resonance theory and Kohonen [19] in self-organizing maps during the last decade, there now exist a vast body of research and researchers in neural networks.

The objectives of this survey are to present the ANN control architectures currently used and being recommended for use by different authors. We summarize the basic learning algorithms that are used to train those architectures in Section 2. Different forms of controller designs are discussed in Section 3. A list of representative applications in the literature is given in Section 4. Finally, some of the trends and needs in development and research are outlined in Section 5.

2. BASIC NEUROCONTROL TOPOLOGIES AND TRAINING

The neural network literature abounds in the paradigms that can be applied to several types of problems. Out of all these encoding algorithms, three classes of encoding algorithms are basic and seem to be the most popular. The first two of the techniques, the backpropagation paradigm and the cerebellar model articulation controller (CMAC) are feedforward nonrecurrent networks. The third network has feedback elements and a well-known version is the Hopfield network.

Backpropagation

Backpropagation [20] is the most popular training algorithm in the training of multilayer artificial neural networks. The learning procedure, called the generalized delta rule, involves the presentation of a set of pairs of input patterns and target output patterns. The system of multilayered networks with a randomly (or otherwise) initialized interconnection weights uses the given input vector to produce its own output vector and compares this with the target output pattern. When there is a difference, the rule for changing weights is given by

$$\Delta_k w_{ji} = \eta \left(t_{kj} - o_{kj}\right) i_{ki} = \eta \delta_{kj},$$

where t_{kj} is the target output for the j^{th} component of the output pattern for pattern k, o_{kj} is the j^{th} element of the actual output pattern produced by the presentation of the input pattern k, i_{ki} is the value of the i^{th} element of the input pattern, $\delta_{kj} = t_{kj} - o_{kj}$ and $\Delta_k w_{ij}$ is the change for the connection strength from i^{th} to j^{th} unit after the k^{th} input vector is used. This gradient-based rule is simple in concept and is the most widely used paradigm.

The method of backpropagation, however, has a few disadvantages: it requires a considerable number of iterations for learning a pattern or a model. For an example of missile control, backpropagation has been found to take about 60,000 iterations. For applications requiring large networks, therefore, on-line training can be time-consuming. Backpropagation requires a large number of computations per iteration which increases with the number of nodes and layers. It may result in error surfaces having relative minima which the gradient-based technique will find hard to overcome. Also, for convergence, the whole input string must be used for weight changes since all layers and nodes are interconnected.

Cerebellar Model Articulation Controller (CMAC)

Albus [21,22] developed the idea of Cerebellar Model Articulation Controller (CMAC) in his pioneering work. CMAC is an adaptive system which operates from a table look-up as opposed to a mathematical solution of equations. In this scheme, the input commands and feedback variables are placed in an input vector which is used to address a memory where the needed output variables are stored. Each address leads to a set of memory locations. The arithmetic sum of the contents in those memory locations is the value of the stored variable.

Perceptron based neural networks, in general, have two mappings. The first mapping f is $f : S \rightarrow A$ where S represents the sensory input vectors and A consists of the association cell vectors. The second mapping g is $g : A \rightarrow P$ where P consists of the response output vectors. As compared to this structure, the CMAC method breaks down the $S \rightarrow A$ mapping into $S \rightarrow M$ and $M \rightarrow A$ mappings.

The $S \rightarrow M$ mapping converts the input vector S to binary variables M. These binary variables then, have an association cell vector A. The output (control) P is the sum of the weights stored in the association cells corresponding to the input S.

The CMAC operates such that the input space consists of the set of all possible input vectors. The number of input vector components and the output vector components are arbitrary and are limited only by practical limits of computations. The first mapping in CMAC carries the analog inputs into a "conceptual" memory by using binary representation. In this mapping, if two inputs are "close" in input space, they will have overlaps in the associated binary representations. As a result, if the proper response is stored for one input, the response will nearly be the same for the second input. This property called "generalization" is similar to the human brain capacity to generalize from one learning experience to the other. On the other hand, if two input patterns are far apart, they will have no overlap in the conceptual memory.

The next step in the working of CMAC is to map the conceptual memory to an actual memory of practical size. The reason is the large number of association cells required in conceptual memory for practical problems. As an example, if a problem has 10 inputs with 10 distinguishable values, 10^{10} cells are required in the conceptual memory. Therefore, methods such as hash coding are used to create a much smaller actual memory. Corresponding to each input, there is an address in the actual memory. The weights are stored as the contents of the memory location. The sum of the weights (contents) of the memory is the output response.

Miller *et al.* [23] propose a control scheme that utilizes the CMAC architecture. An on-line learning scheme is used to adjust the CMAC memory. The learning process consists of determining the memory content values in the practical memory to represent the function being learnt over the relevant input space. Let the observed value of the function be F_0 where s_0 is the given input. If the corresponding computed output of the CMAC module is f_0, then the correcting factor, δ, is determined from

$$\delta = \frac{\beta\,(F_0 - f_0)}{c},$$

where β is a training or learning rate factor such that $0 < \beta < 1$ and c is the number of locations in the practical memory to which each input s_0 maps.

The CMAC possesses a lot of desirable properties. It can accept analog inputs and produce analog outputs. CMAC has built in generalization whereby the "close" input vectors map into output vectors that are "close" even if the input was not part of the training set, but were contained in a training region. Even large CMAC networks can be trained in a small number of iterations. (For the same pattern recognition problem, Miller *et al.* state that CMAC took 50 iterations while a two-layer backpropagated perceptron network took 12,000 iterations.) CMAC can learn a wide variety of functions and has a practical hardware realization.

An important distinction between the backpropagation technique and the CMAC is that every input pattern affects every association cell in the former whereas in the CMAC, it affects only a limited number 'close' to it. This feature may help 'generalization' better while inhibiting 'interference.' The main drawback of CMAC is that it is not easily amenable to mathematical stability and performance analysis.

Hopfield Network

Backpropagation and CMAC algorithms are nonrecurrent. The absence of feedback implies stable behavior, however, it also means that some valuable information is not used. Hopfield network, the third important paradigm, is a recurrent network. It consists of multilayers and nodes with feedback paths from their outputs back to their inputs. For stable networks, the resultant outputs reach steady state values.

The single layered Hopfield method has lateral and recurrent connections. The input vectors a_i of k^{th} pattern feed into the layer and can be output anytime with the equation

$$c_i a_i = \sum_{j=1}^{n} f\,(a_j)\, w_{ji} - \frac{a_i}{R_i} + I_i,$$

where c_i is a positive constant representing the i^{th} processing elements input capacitance, R_i is a positive constant, I_i is the external input to the i^{th} processing element, $f(\cdot)$ is a sigmoid or threshold function, and w_{ji} is the symmetric connection between j^{th} to i^{th} processing element.

Sufficient conditions for the stability of this system have been shown through the formulation of an energy function. In solving a complex problem with the Hopfield network, the formulation of the energy function and the associated weights takes a major effort.

3. APPROACHES TO ARTIFICIAL NEURAL NETWORK CONTROLLER DESIGN

The task of a control system designer is to devise control laws which direct a given system to stated goals in a specified manner. This task requires the knowledge of the system that the controller is trying to influence and the form of the controller. In this section, five basic approaches [24] to designing a neural net controller are presented. Some of these assume explicit knowledge of the plant to be controlled.

Supervised Control

In this mode, the neural controller learns to reproduce a known task. Such an approach requires an input vector and a corresponding output vector. In an actual situation, it can be a robot controller learning to produce a proper control in response to inputs which are sensor readings. The supervised control can be implemented through many types of supervised learning methods.

Direct Inverse Control

In this mode, the neurocontroller is made to learn the mapping from the state space back to the control signals. The trajectories of the systems are the input to the controller and the outputs are the desired controls. The controller can be trained off-line with nominal trajectories either through simulation or through actual system performance. If there is no one-to-one mapping between the spatial coordinates and actuator signals, an alternative design is to combine it with a forward neural network model of the plant. Any supervised learning method can be used as a training algorithm.

Backpropagation Through Time

In this design of the controller, the neural network uses a loss function at the final time that it needs to optimize. This approach combines a model of the plant environment with the controller. Backpropagation technique is used to calculate the derivatives of the cost function at the terminal time with respect to the current controller actions. These derivatives are used in the weight training of the networks.

Adaptive Critic

This class of designs consists of an inverse controller called an action network and a network called critic which simulates a performance index and which criticizes or evaluates the outputs of the action network. In addition, this configuration may include a model of the environment. A functional is chosen as a critic element. Many cost functionals are usually dependent on the states as well as the control and are driven by the minimizing control. Consequently, if the controller is nonminimizing (the output of the neurocontroller), it will result in a nonminimum cost functional.

At each step, the derivatives of the cost functional are backpropagated through the environment model to compute the desired control. The difference between the desired output and the controller output is used to adjust the connectionist weights of the action network. In a variant of the adaptive critic [25], the critic output is used to 'reward or punish' the controller directly through modification of its weights.

Neural Adaptive Control

In neural adaptive control, the input to the neural controller comes from a trajectory created by a reference model desired by the designer. With a model-reference adaptive controller, the neurocontroller, the ANN must learn to make the plant follow the path specified by the reference model.

4. APPLICATIONS OF NEURAL NETWORKS TO CONTROL OF DYNAMICAL SYSTEMS

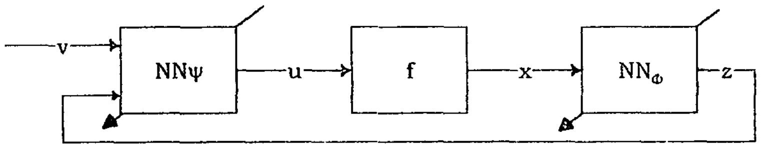
As the problems in engineering become more complex, we seek new means and techniques to solve them. Since the neural networks are massively parallel in nature, they can perform computations faster and help find solutions in computation-intensive problems. They can be adapted to a changing environment and can have nonlinear representations. Physical situations such as manufacturing processes and their control can be represented through neural networks in a more straight-forward fashion, whereas rigorous mathematical models are difficult to develop. Consequently, a variety of control applications have been reported in the literature. Long standing fascination of human beings with artificial intelligence and autonomous systems has resulted in a majority of these applications being in the areas of robotics and automation.

A major weakness in the application of neural networks in control is the lack of rigorous theoretical results. Results to date are mainly experimental and anecdotal. Though beyond the scope of this survey, even theoretical work pertaining to the capabilities of artificial neural networks is lacking. Hornik, Stinchcombe, and White [26] (among others) have shown that artificial neural networks can approximate an arbitrary continuous function to any degree of accuracy given enough neurons in a single hidden layer. Some remaining open questions include, the actual number of neurons needed, type and convergence of weight setting algorithms, and effect of the number of layers.

Narenda and Levin [27] have developed one of the few theoretical frameworks for using neural networks for controllability and stabilization of dynamical systems. Their work is restricted to multilayer feedforward networks using Dynamic Back Propagation and nonlinear systems with complete access to state information. Given a system of the form:

$$x(k+1) = f[x(k), u(k)],$$

where $x(k) \in \chi \subset \Re^n$, $u(k) \in \mathcal{U} \subset \Re^r$, and $f(0,0) = 0$ so that $x = 0$ is an equilibrium, conditions are given under which two neural networks $u = NN_\Psi(\nu, z)$ and $z = NN_\Phi(x)$ $(z \in \Re^n, \nu \in \Re^r)$ can be trained to feedback linearize and stabilize the system. The architecture is shown below.



Results are extended to systems that are not feedback linearizable. If the controllability matrix around the origin is of full rank, then a methodology and conditions are developed under which a single neural network can be trained to directly stabilize the system around the origin.

Sontag [28] develops results in which he shows that, in general, nonlinear systems can be stabilized using neural networks containing two hidden layers. The result opposes intuitively the neural net approximation theories, which indicate that single hidden layer networks are universal approximators. The formulation of the control problem as an inverse kinematics problem, not an approximation problem, is the basis of Sontag's results.

Barto [29] terms the research on neural networks as connectionist research and makes an interesting comparison between the connectionist learning methods to those studied in the established area of traditional adaptive control. If a parameter estimation problem has to be solved using neural networks, the representations are often chosen from the perception of how nervous systems represent information. As against this, in a conventional approach, physics of the problem (with very little help from theory) is used to make decisions on problem representation. The rules for decision making or modelling in a conventional approach relate to the selection of the number of parameters and complexity and, in a connectionist approach, it depends on the structure of the network and the interrelation between the connectionist weights. The use of *a priori* knowledge can be easily incorporated in a conventional controller, whereas in a neural network, it is usually an input output type relationship.

The performance evaluation in both approaches can be done through cost functions such as least mean squared error. In both conventional and connectionist methods, we can have on-line and off-line methods. In off-line methods, all the training data are available at one time. In on-line methods (for use in missile guidance, as an example), however, the needed feature is learn-as-you-go and, therefore, the methods must be very efficient so as to keep up with the change of events with time.

According to Narendra, controllers based on neural networks will be effective in four scenarios. The first case is design of a controller for a nonlinear plant where analytical formulation will be too complex. The second use is the identification of the plant, such as with flexible space structures, can be better accomplished with a neural network. The third case is in the area of process control where the plant has to operate at several set points. The fourth case is a situation where the state space is partitioned into disjoint regions, and the neural network can be used to generate optimal control corresponding to each region.

Nguyen and Widrow [30] demonstrated the power of backpropogation through time for nonlinear control with the "celebrated" Truck Backer-Upper. Their approach first trained a neural network to model a truck tractor trailer. The model was then imbedded in a backpropogation through time scheme that developed a controller that could back the tractor trailer up to a loading dock. This problem has become a benchmark in fuzzy and neural control.

Miller *et al.* [31–35] discuss a general learning algorithm for CMAC and its application to the dynamic control of robotic manipulators. In this application, the dynamics are not required to be known. The control scheme learns the process through input and output measurements. Through the results from applications to robots, Miller *et al.* conclude that

(1) the CMAC learning control provides better performance as compared to the fixed-gain controllers,
(2) since measured and estimated values must be converted to discrete form, the resolution and range allowed for each variable need to be selected carefully, and
(3) the most sensitive design parameter is the number of memory locations addressed by each input state in the CMAC architecture.

Yamamura *et al.* [36] compare two methods of training in a neural network control of a two link manipulator to draw characters. Their modified backpropagation method has the following terms which reduce the magnitude of all weights and the weights connected to less active neurons as

$$E_1 = \sum_i \log\left[\cosh\left(p_i\right)\right],$$

$$E_2 = \sum_{j \prec i} S_i S_j,$$

where

$$S_i = \sum_j \sigma\left(W_{1i_j}\right) \sum_k \sigma\left(W_{1k_i}\right),$$

and

$$\sigma(p) = \frac{p^2}{1+p}.$$

In trying to draw the letter 'm', they show that a backpropagation method with the new training terms draws better than with the standard training.

Liu, Iberall, and Bekey [37] examine the question of developing device-dependent robot hand controllers. Their argument for such a controller is that it would allow the low level control problems to be separated from high level functionality. In order to accomplish this objective, they use a backpropagation algorithm with one hidden layer consisting of four neurons. The inputs are defined by dimensions of the object and the outputs are defined by the grasp modes. In this manner, they have shown through simulation that they are able to construct a p-g table.

Miller *et al.* [38] state that they use CMAC in the real time control of an industrial robot and in other applications. They use hundreds of thousands of adjustable weights in their networks which, in their experience, converge in a small number of iterations.

Wang and Yeh [39] use a backpropagation architecture in the control of a robot model which simulates PUMA560. Their self-adaptive neural controller (SANC) consists of a network to model the plant and a controller network. The plant model is trained either off-line with the outputs of a mathematical model or on-line with outputs of the plant through excitations. In the "controlling and adapting" phase, the control network is adapted by working in series with the plant network. The other feature of this training is that the control network is also trained off-line in a "memorizing phase" with results from the adapting phase in a random manner. According to the authors, this feature helps overcome the temporal instability inherent in backpropagation. Their numerical results indicate that the SANC procedure yields good trajectory-following accuracy.

Lee and Kil [40] use a structure which has a multilayer feedforward network with hidden units having sigmoidal activation functions and a feedback network forming a recurrent loop around the feedforward network. They use the feedforward network to represent the forward kinematics of a robotic arm. They obtain the feedback network to iteratively generate input joint angle correction terms based on a Liapunov-function until the output of the feedforward network converges to the desired Cartesian position. They present several simulations that show that the end point of the given initial joint configuration converges to the desired position.

Guez and Ahmad [41] use a backpropagation network to solve a nonlinear inverse kinematic problem arising in the control of robotic manipulators. They conclude, through their numerical simulations, that the trained neural networks can give good initial guesses that can be used in the iterative solutions to the nonlinear inverse kinematic problems. A good initial guess, in turn, will result in a faster convergence to the solution, and thus will help with achieving real time operations of the robots.

Jamshidi *et al.* [42] report a case study of a neural network-based controller for a two-link robot. They use a two-layer perceptron with a sigmoidal function in the first layer and a scaling in the second layer. Their study is not conclusive on the training sampler or iterations or the superiority of the neural network over the conventional controllers. However, they feel that a multilayer perceptron network is extremely good in approximating the Euler-Lagrange equations of a robot.

Karakasoglu and Sundareshan [43] use a backpropagation neural net for a decentralized, variable structure control of robotic manipulators. The variable structure control is used by virtue of its insensitivity to parameter variations and disturbances during the sliding mode. They use the neural network to generate the control to offset the deviations from the sliding line. Karakasoglu *et al.* demonstrate with a Stanford arm movement that the neural network based controller works very well.

In a paper by Anderson [44], neural network learning methods which learn to generate action sequences by using two functions are described. The first one, called an action function, maps the current state into control actions. The second one is called an evaluation function or a critic and it maps the current state into an evaluation of that state. The network that learns the action is called an action network and the network that learns the evaluation on the individual actions is called the evaluation network. This is a variant of an adaptive critic architecture.

Lin and Kim [45] use the CMAC technique to solve the "box-based" adaptive critic scheme of Barto *et al.* Their numerical experiments on the inverted pendulum problem show that CMAC is more efficient than the box-type method since the CMAC scheme can generalize better. They show that the critical region before the fall is small and therefore, memory size is not a problem, unlike a problem with a large input vector.

Jameson [46] uses a neurocontroller with a backpropagated adaptive critic to balance a pole from falling in a pole-cart problem. While a model network is used in a basic adaptive critic design, Jameson uses a "delta-state network" where the output is $k(x_{t+1} - x_t)$ where k is a scale factor, since the change in state is more relevant to the problem than the state x_t at time t. His critic network's output is maximized, as suggested by Werbos [47], by feeding the gradient back to the action network. However, he uses the expression for the goal as Barto *et al.* Jameson's numerical experiments show that his variant of the architecture is faster but more unstable when compared to Anderson's work.

Moore [48] uses reinforcement learning in the control of a nonlinear system. He describes a multilayer neural network with real-valued outputs which learns by using a combination of reinforcement learning and backpropagation. His control input is a normal random variable with the output of the action network as the mean and the predicted reinforcement being the standard deviation. Depending on the standard deviation, the control action taken is closer to the mean or away from it. Applied to a signal tracking problem, Moore's reinforcement scheme seems to perform well.

Sofge and White [49] use a real-time direct-inverse neurocontrol system for controlling the manufacture of thermoplastic composite structures. Their network architecture consists of a neural controller and a second neural network which adaptively criticizes the performance of the controller based on utility functions. Their experiments indicate that the CMAC architectures show rapid convergence characteristics as compared to the backpropagation method but are very sensitive to scaling of derivative terms in the critic. They also show that the critic with the derivative of the performance index performs much better than the architecture without the derivative terms.

Werntges [50] uses an error gradient based on a cost function as a critic. This critic is used as an interface between a backpropagation neural network and a robot arm. Application to a two-joint robot arm shows that the critic based network learns the differential inverse kinematics at least as well as a supervised training based network.

Iiguni *et al.* [51] present a nonlinear regulator design method for nonlinear systems in the presence of system uncertainties with multilayered neural networks. This regulator consists of a network for identification of the plant and a second network for control. Their numerical experiments show that the single network based architecture with a controller performs much worse in time as compared to the two-network design. The two-network based design is shown to be able to stabilize an unstable plant in cases where the classical linear optimal regulator fails.

Kehtarnavaz and Sohn [52] use a backpropagation network to learn the steering function for use in autonomous vehicles. Movement of autonomous vehicles consists of path planning and controls. The difficulty with path planning is the need for quick conversion of the sensor information such as heading angles and range into appropriate steering angles which are nonlinear functions of the information. Kehtarnavaz and Sohn present numerical results with various backpropagation networks which show the feasibility of using neural networks for computing steering functions for autonomous guidance.

Steck and Balakrishnan [53] use a Hopfield network to solve an optimal guidance problem. By reformulating the linear quadratic performance index as a Hopfield energy function, they solve for the control accelerations at each instant for a homing missile intercept. They show that the relative range between the missile and the target goes to zero in different target intercept engagements.

Tsutsumi and Matsumoto [54] present a new method for a robotic manipulator position control. The manipulator is modeled by using the distances between the joint locations. By placing the geometric constraints through energy functions, they show that the guidance of manipulator end effector can be converted to a Hopfield energy function. Through simulations, their converged network results show that the end effector can be moved successfully to a target point. Tsutsumi *et al.* extend their previous results from a manipulator with rigid arms to one with an elastic arm. They also demonstrate that this method can be used for the configuration control of a two-dimensional truss. Xu *et al.* [55] have used a Hopfield network to optimally place a three-finger robot gripper system.

Tolat and Widrow [56] describe the use of a single linear adaptive threshold element (ADALINE) is keeping an inverted pendulum stable. Their simulations demonstrate that their network is able to control the pendulum with time sequences of visual images as inputs.

Wang and Miu's [57] adaptive neural network control system consists of a controller network and an identifier network which is a plant emulator. They have two learning mechanisms which are implemented through backpropagation. The identifier network learns through the difference between its output and the actual states. The controller learns through minimizing the cost function based on the estimated output and the desired output. Wang and Miu's numerical experiments from an inverted pendulum problem illustrate the good learning capabilities of their controller.

Elsley [58] presents a control architecture which autonomously learns to perform kinematic control of an unknown system. The training phase in Elsley's paper is described as follows: the motion of the system due to a previous command is given to the controller as the desired motion. The difference between the output of the controller and the earlier command that produced the current state is backpropagated to adjust the weights. Such an inverse control, Elsley feels, requires a large number of neurons. Therefore, he replaces the inverse controller with an inverse Jacobian network control where the Jacobian represents the derivative of the system transfer function with respect to the controls. This neurocontroller is shown to adapt to changes in the system and the controller. Furthermore, it is shown to be fault tolerant with considerable (30%) component failures.

Campagna and Kraft [59–61] compare a CMAC neural network to a self-tuning controller and a Liapunov based model reference controller. The example applications include a linear system with noisy and noise-free measurements and a nonlinear system. According to their findings, the self-tuning regulator learns the unknown system parameters very quickly and handles noisy measurements well. However, it does not track the nonlinear system well. The model reference controller is slow to converge even in the noise-free case and shows instability with the noise. The neural network takes the longest time to learn the system. However, change of the reference input signal is found not to affect the performance. Also, the neural network approach is shown to perform very well with a nonlinear system.

Narendra and Parthasarathy [62] develop different identification and controller structures using neural networks. They argue that the multilayer feedforward networks and recurrent networks are similar, although the multilayer networks are representative of static nonlinear maps and the recurrent networks represent nonlinear dynamic feedback systems. They have present four network models of different complexities for the identification and control of nonlinear dynamical systems and demonstrated through simple examples.

Wilmhelmsen and Cotter [63] compare three different neural network architectures for the control of a single-degree-of-freedom robotic arm whose motion is nonlinear. They use

(1) a neural network as an inverse model,
(2) a linear controller and an ANN based feedforward controller, and
(3) an ANN with a recurrent neural network (backpropagation in time).

Their simulations show that the first architecture leads to instability while the second and the third perform well within limits of accuracy.

Guez and Selinsky [64] propose a supervised training scheme for a cart-pole system where the supervised training commands are carried out by a human being as a teacher. Guez *et al.* term such a neurocontroller as Trainable Adaptive Controllers. Their numerical experiments with a neurocontroller trained with a human teacher with and without a first-order delay for the human reactions show that such TAC's perform well in controlling an unstable plant.

Narendra and Mukhopadhyay [65] treat a fault detection and control problem through 'intelligent control.' When there is a fault, any system changes its configuration. Corresponding to each altered state, Narendra *et al.* assume that a controller which produces a desired response of the system exists. Their neural network architecture consists of a neural network that recognizes the patterns of output to indicate the configuration of the system and two networks at a lower level to identify the plant and produce a control action. They provide some numerical simulations to illustrate their ideas.

Narendra and Levin [66] address the problem of regulation of nonlinear dynamical systems which have multiple equilibrium states. The neural networks lack accuracy in their mapping capabilities if the inputs are not uniformly distributed in the region of interest. Hence, regions of states of nonlinear systems where unstable points exist pose a problem. In order to obviate this problem, Narendra and Levin recommend use of multiple neural networks.

Berenji [67] combines the fields of artificial neural networks and fuzzy logic to propose an approximate reasoning-based intelligent control (ARIC). His ARIC consists of an action-state evaluation network which acts as a critic and an action selection network which has a fuzzy controller. In using both elements, he feels that we can fuse a human understandable expression of the knowledge used in fuzzy control rules and the adjustment of the performance by a neural network which learns by experience. He suggests many examples in space such as rendezvous and docking, proximity camera-tracking systems and tethered systems control where use of ARIC can be beneficial.

Lane *et al.* [68] introduce B-spline (basis splines) receptive field function with more general CMAC weight addressing schemes help develop higher-order CMAC networks. These CMAC networks are capable of learning not only the functions but also the function derivatives. For relatively small problem sizes (less than four inputs), their experiments show that B-spline CMAC's learn functions very fast and perform very accurately.

Carter *et al.* [69] discuss the fault tolerance characteristics of CMAC network in their study. They investigate the effects of faults in an adjustable weight layer which has been trained. They study the degradation of approximation with an application where the network learns functions of a single variable. They conclude, after numerical experiments, that the CMAC network fault sensitivity is not uniform and that the location of the sensitive weights is dependent on the function to be learned. They suggest that the enhancement of fault tolerance by increasing the generalization with the distribution of the response computations over more weight locations is undesirable because it increases the sensitivity of faults due to weights that are saturated.

Parisini and Zoppoli [70] propose a neural control architecture based on what has been called the "linear-structure preserving principle." They are able to design trajectory tracking controllers for nonlinear dynamics systems. Control strategies are restricted to the structure of multilayer feedforward networks. The problem is reduced to nonlinear programming problems solved via backpropogation.

Hoskins *et al.* [71] use a constrained iterative inversion technique. A neural net is used to learn a forward model of the plant. Iterative inversion is then performed online to generate control commands. They suggest their method allows the controllers to adapt on-line to changes in the plant dynamics. The method also avoids the analysis difficulty when nonlinear neural networks are introduced into the feedback path. The technique is extended to neural network-based model reference adaptive control, for systems having significant complexity between control inputs and observed output.

Sebald and Schlenzig [72] explore the use of neural nets for the control of highly uncertain systems. Their methodology uses evolutionary programming (EP) and a modified CMAC architecture to regulate systems with unknown time delays. Design of a neural controller is viewed as a game with a minimax criterion to optimize performance across the worst case plant. Results yield the best choice of neural control parameters as well as the plant that is most difficult for the controller to handle. Practical examples include drug infusion and HVAC/lighting/utility systems.

Sastry *et al.* [73], investigate a form of recurrent neural networks termed "memory neuron networks" in identification and adaptive control of nonlinear systems. This form of neural network is formed by augmenting feedforward networks with temporal elements. The advantage of this network structure is the ability to identify dynamical systems without explicit past input and output history. Additionally, the networks can identify systems with both unknown order and time delays. The methodology's justification is presented through preliminary theoretical analysis, and extensive simulation.

5. CLOSURE

The interest in neurocontrol is growing rapidly. At every technical conference, new applications are being introduced and the advantages of using neural networks over other traditional methods are presented. However, theories to explain the good results, especially concerning stability issues, are not as advanced.

In an article which incorporates the building blocks for the neural network structure, Narendra [74] further feels that the analysis stability of the systems modeled through the neural networks has not been undertaken extensively. He feels that the stability theories built around the linearization of systems will not be adequate to fully describe the behavior of the neural networks and that new concepts and techniques are required to deal with large uncertainties that the controllers might face in practical applications.

An important and promising area for exploration is failure-tolerant control based on failure isolation. In many critical areas such as aircraft flights and nuclear reactor operations, it is not enough for a controller to perform the primary design goal which is to assure stable and satisfactory operation of the plant. There is a need (or tremendous loss in its absence) to operate the plant acceptably when faults occur. Under such faulty conditions impending or which have recently occurred, neural networks can be very helpful. Satisfactory modeling of a nonlinear plant under uncertain faulty conditions is extremely difficult. A unique feature of the neural networks, however, is that it can learn complex nonlinear relationships and models. Reliable networks to accomplish such tasks are yet to be developed and analyzed.

The NSF workshop on Aerospace Applications of Neurocontrol [75] lists the need to look at the advantages and disadvantages of hetrarchical and hierarchical structures and the integration of knowledge based representation with neural network techniques as promising areas of future research.

REFERENCES

1. P.K. Simpson, *Artificial Neural Systems*, Pergamon Press, Elmsford, NY, (1990).
2. W.S. McCulloch and W. Pitts, A logical calculus of the ideas immanent in nervous activity, *Bulletin of Math. Biophysics* **5**, 115–133 (1943).
3. F. Rosenblatt, The Perceptron, a probabilistic model for information storage and organization in the brain, *Psychology Review* **62**, 386–408 (1958).
4. F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, Spartan Books, Washington, DC, (1962).

5. F. Rosenblatt, A bibliography of perceptron literature, Collected Technical Papers Vol. 2, Cognitive Systems Research Program, Report No. 4, Cornell University, (July 1963).
6. B. Widrow and M. Hoff, Adaptive switching circuits, WESCON Convention Record: Part 4, pp. 96–104, (1960).
7. B. Widrow, *Generalization and Information Storage in Networks of Adaline 'Neurons' Self-Organizing Systems 1962*, (Edited by M. Yovitz, G. Jacobi and G. Goldstein), pp. 435–461, Spartan Books, Washington, DC, (1962).
8. B. Widrow, An adaptive 'Adaline' neuron using chemical memistors, Stanford Electronics Laboratory, Technical Report, 1553-2, (1959).
9. B. Widrow, Adaline and Madaline—1963, Plenary speech, In *Proceedings 1st IEEE International Conference on Neural Networks*, San Diego, CA, June 23, 1987, Vol. 1, pp. 145–158.
10. B. Widrow, N. Gupta and S. Maitra, Punish/reward: Learning with a critic in adaptive threshold systems, *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-3** (5) (September 1973).
11. M.L. Minsky and S.A. Papert, *Perceptrons: An Introduction to Computational Geometry*, Expanded edition, MIT Press, Cambridge, MA, (1988).
12. P.J. Werbos, Beyond regression: New tools for prediction and analysis in the behavioral sciences, Ph.D. Thesis, Harvard University, Cambridge, MA, (August 1974).
13. D.E. Rumelhart, G.E. Hinton and R. Williams, Learning internal representations by error propagation, In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, (Edited by D.E. Rumelhart, J.L. McClelland and PDP Research Group), MIT Press, Cambridge, MA, (1986).
14. S.E. Dreyfus, Artificial neural networks, back propagation, and the Kelley-Bryson gradient procedure, *Journal of Guidance, Control, and Dynamics* **13** (5) (September/October 1990).
15. J.J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, In *Proceedings National Academy of Science*, April 1982, Vol. 79, pp. 2554–2558.
16. J.J. Hopfield, Neurons with graded response have collective computational properties like those of two-state neurons, In *Proceedings National Academy of Science*, May 1984, Vol. 81, pp. 3088–3092.
17. G.A. Carpenter and S. Grossberg, A massively parallel architecture for a self-organizing neural pattern recognition machine, *Computer Vision, Graphics, and Image Processing* **37**, 54–115 (1987).
18. G.A. Carpenter and S. Grossberg, ART-2: Self-organization of stable category recognition codes for analog input patterns, *Applied Optics* **26**, 4919–4930 (1987).
19. T. Kohonen, *Self-Organization and Associative Memory*, 2nd edition, Springer-Verlag, New York, (1988).
20. D.E. Rumelhart, G.E. Hinton and R. Williams, Learning internal representations by error propagation, In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, (Edited by D.E. Rumelhart, J.L. McClelland and PDP Research Group), MIT Press, Cambridge, MA, (1986).
21. J.S. Albus, A new approach to manipulator control: The Cerebellar Model Articulation Controller (CMAC), *Transactions of the ASME*, 220–227 (September 1975).
22. J.S. Albus, Data storage in the Cerebellar Model Articulation Controller, *Transactions of the ASME*, 228–233 (September 1975).
23. W.T. Miller, F.H. Glanz and L.G. Kraft, CMAC: An associative neural network alternative to backpropagation, *Proceedings of the IEEE* **78** (10), 1561–1567 (October 1990).
24. P.J. Werbos, Backpropagation through time: What it does and how to do it, *Proceedings of the IEEE* **78** (10), 1550–1560 (October 1990).
25. A.G. Barto, R.S. Sutton and C.W. Anderson, Neuronlike elements that can solve difficult learning control problems, *IEEE Trans. on System, Man, and Cybernetics* **13**, 835–846 (1983).
26. K. Hornik, M. Stinchcombe and H. White, Multilayer feedforward networks are universal approximators, *Neural Networks* **2** (5), 359–366 (1989).
27. K.S. Narendra and A.V. Levin, Control of nonlinear dynamical systems using neural networks: Controllability and stabilization, *IEEE Transactions on Neural Networks* **4** (2), 192–206 (March 1993).
28. E.D. Sontag, Feedback stabilization using two-hidden-layer nets, *IEEE Transactions on Neural Networks* **3** (6), 981–990 (November 1992).
29. A.G. Barto, Connectionist learning for control: An overview, In *Neural Networks for Control*, (Edited by W. Miller, III, R.S. Sutton and P. Werbos), MIT Press, (1990).
30. D. Nguyen and B. Widrow, The truck backer-upper: An example of self-learning in Neural Networks, In *Neural Networks in Control*, (Edited by W.T. Miller, R. Sutton and P. Werbos), MIT Press, Cambridge, MA, (1990).
31. W.T. Miller, F.H. Glanz and L.G. Kraft, Application of a general learning algorithm to the control of robotic manipulators, *International Journal of Robotics Research* **6**, 84–98 (Summer 1987).
32. W.T. Miller, Sensor-based control of robotic manipulators using a general learning algorithm, *IEEE J. Robotics Automat.* **RA-3**, 157–165 (April 1987).
33. W.T. Miller, Real time learned sensor processing and motor control for a robot with vision, In *1st Annual Conference of the International Neural Network Society*, September 1988.
34. W.T. Miller, Real time application of neural networks for sensor-based control of robots with vision, *IEEE Transactions on System, Man, and Cybernetics* **19**, 825–831 (July/August 1989).
35. W.T. Miller and R.P. Hewes, Real time experiments in neural network based learning during high speed nonrepetitive robotic operations, In *Proceedings 3rd IEEE Int. Symposium on Intelligent Control*, August 24–26, 1988, pp. 513–518.

36. A.A. Yamamura, A. Sideris, C. Ji and D. Psaltis, Neural network control of a two-link manipulator, In *Proceedings of the 29th Conference on Decision and Control*, Honolulu, HI, December 1990, pp. 3265–3266.
37. H. Liu, T. Iberall and G.A. Bekey, Building a generic architecture for robot hand control, In *IEEE International Conference on Neural Networks*, San Diego, CA, July 24–27, 1988, Vol. II.
38. W.T. Miller, F.H. Glanz and L.G. Kraft, CMAC: An associative neural network alternative to backpropagation, *Proceedings of the IEEE* **78** (10), 1561–1567 (October 1990).
39. S.D. Wang and M.S.H. Yeh, Self-adaptive neural architectures for control applications, In *International Joint Conference on Neural Networks*, San Diego, CA, June 17–21, 1991, Vol. III, pp. 309–314.
40. S. Lee and R. Kil, Robot kinematic control based on bidirectional mapping neural network, In *International Joint Conference on Neural Networks*, San Diego, CA, June 17–21, 1991, Vol. III, pp. 327–336.
41. A. Guez and Z. Ahmad, Solution to the inverse kinematics problem in robotics by neural networks, In *IEEE International Conference on Neural Networks*, San Diego, CA, July 24–27, 1988, Vol. II.
42. M. Jamshidi, B. Horne and V. Nader, A neural network-based controller for a two-link robot, In *Proceedings of the 29th Conference on Decision and Control*, Honolulu, HI, December 1990, pp. 3256–3257.
43. A. Karakasoglu and M.K. Sundareshan, Decentralized variable structure control of robotic manipulators: Neural computational algorithms, In *Proceedings of the 29th Conference on Decision and Control*, Honolulu, HI, December 1990, pp. 3258–3259.
44. C.W. Anderson, Learning to control an inverted pendulum using neural networks, *IEEE Control Systems Magazine*, 31–37 (April 1989).
45. C.S. Lin and H. Kim, CMAC-based adaptive critic self-learning control, *IEEE Transactions on Neural Networks* **2** (5) (September 1991).
46. J. Jameson, A neurocontroller based on model feedback and the adaptive heuristic critic, In *1990 International Joint Conference on Neural Networks*, San Diego, CA June 17–21, 1990.
47. P.J. Werbos, Neural networks for control and system identification, In *Proceedings of the 28th Conference on Decision and Control*, Tampa, FL, December 1989, pp. 260–265.
48. K.L. Moore, A reinforcement-learning neural network for the control of nonlinear systems, In *1991 American Control Conference*, Boston, MA, June 26–28, 1991, pp. 21–24.
49. D.A. Sofge and D.A. White, Neural network based process optimization and control, In *Proceedings of the 29th Conference on Decision and Control*, Honolulu, HI, December 1990, pp. 3270–3276.
50. H. Werntges, Delta rule based neural networks for inverse kinematics: Error gradient reconstruction replaces the teacher, In *International Joint Conference on Neural Networks*, San Diego, CA, June 17–21, 1991, Vol. III, pp. 415–420.
51. Y. Iiguni, H. Sakai and H. Tokumaru, A regulator design method using multi-layered neural networks, In *International Joint Conference on Neural Networks*, San Diego, CA, June 17–19, 1990, Vol. III, pp. 371–380.
52. N. Kehtarnavaz and W. Sohn, Steering control of autonomous vehicles by neural networks, In *1991 American Control Conference*, Boston, MA, June 26–28, 1991, pp. 3096–3101.
53. J. Steck and S.N. Balakrishnan, Use of Hopfield neural networks in optimal guidance, In *29th Aerospace Sciences Meeting*, Reno, NV, January 7–10, 1991.
54. K. Tsutsumi and H. Matsumoto, Neural computation and learning strategy, In *Proceedings of IEEE First Annual International Conference on Neural Networks*, Vol. IV, pp. 525–534, (1987).
55. G. Xu, H.K. Scherrer and H.S. Schweitzer, Application of neural networks on robot grippers, In *International Joint Conference on Neural Networks*, San Diego, CA, June 17–21, 1991, Vol. III, pp. 337–342.
56. V.V. Tolat and B. Widrow, An adaptive 'broom balancer' with visual inputs, In *IEEE International Conference on Neural Networks*, San Diego, CA, July 24–27, 1988, Vol. II.
57. G.J. Wang and D.K. Miu, Unsupervised adaptive neural-network control of complex mechanical systems, In *1991 American Control Conference*, Boston, MA, June 26–28, 1991, pp. 28–31.
58. R.K. Elsley, A learning architecture for control based on backpropagation neural networks, In *IEEE International Conference on Neural Networks*, San Diego, CA, July 24–27, 1988, Vol. II.
59. D.P. Campagna and L.G. Kraft, III, Comparison of memory update algorithms for CMAC neural network adaptive control systems, In *Proceedings of the 28th IEEE Conference on Decision and Control*, Tampa, FL, December 1989.
60. G. Kraft and D.P. Campagna, A comparison of CMAC neural network and traditional adaptive control systems, In *Proceedings 1989 American Control Conference*, June 21–23, 1989, Vol. 1, pp. 884–889.
61. L.G. Kraft, III and D.P. Campagna, Comparison of convergence properties of CMAC neural network and traditional adaptive controllers, In *Proceedings of the 28th IEEE Conference on Decision and Control*, Tampa, FL, December 1989.
62. K.S. Narendra and K. Parthasarathy, Identification and control of dynamical systems using neural networks, *IEEE Transactions on Neural Networks* 1 (1), 4–27 (March 1990).
63. K. Wilhelmsen and N. Cotter, Neural network based controllers for a single degree of freedom robotic arm, In *International Joint Conference on Neural Networks*, San Diego, CA, June 17–21, 1991, Vol. II, pp. 407–413.
64. A. Guez and J. Selinsky, A neuromorphic controller with a human teacher, In *IEEE International Conference on Neural Networks*, San Diego, CA, July 24–27, 1988, Vol. II.
65. K.S. Narendra and S. Mukhopadhyay, Intelligent control using neural networks, In *1991 American Control Conference*, Boston, MA, June 26–28, 1991, pp. 1069–1074.
66. K.S. Narendra and A.V. Levin, Regulation of nonlinear dynamical systems using multiple neural networks, In *1991 American Control Conference*, Boston, MA, June 26–28, 1991, pp. 1609–1614.

67. H.R. Berenji, Artificial neural networks and approximate reasoning for intelligent control in space, In *1991 American Control Conference*, Boston, MA, June 26–28, 1991, pp. 1075–1080.
68. S.H. Lane, D.A. Handleman and J.J. Gelfand, Higher-order CMAC neural networks—Theory and practice, In *1991 American Control Conference*, Boston, MA, June 26–28, 1991, pp. 1579–1585.
69. M.J. Carter, F.J. Rudolph and A.J. Nucci, Operational fault tolerance of CMAC networks, In *Advances in Neural Information Processing Systems 2*, (Edited by D.S. Towretzky), pp. 340–347, Morgan Kaufmann, San Mateo, CA, (1990).
70. T. Parsini and R. Zoppoli, Neural networks for feedback feedforward nonlinear control systems, *IEEE Transactions on Neural Networks* **5** (3), 436–449 (May 1994).
71. D.A. Hoskins, J.N. Hwang and J. Vagners, Iterative inversion of neural networks and its application to adaptive control, *IEEE Transactions on Neural Networks* **3** (2), 292–301 (March 1992).
72. A.V. Sebald and J. Schlenzig, Minimax design of neural net controllers for highly uncertain plants, *IEEE Transactions on Neural Networks* **5** (1), 73–82 (January 1994).
73. P.S. Sastry, G. Santharam and K.P. Unnikrishnan, Memory neuron networks for identification and control of dynamical systems, *IEEE Transactions on Neural Networks* **5** (2), 306–319 (March 1994).
74. K.S. Narendra, Adaptive control using neural networks, In *Neural Networks for Control*, (Edited by W.T. Miller, R. Sutton and P. Werbos), MIT Press, Cambridge, MA, (1990).
75. D. Sofge and D. White, NSF workshop on aerospace applications of neurocontrol, *IEEE Control Systems Magazine* (April 1991).

ADDITIONAL REFERENCES

Antsaklis, P.J., Neural networks for control systems, *IEEE Transactions on Neural Networks* **1** (2), 242–244 (June 1990).

Brown, M., C.J. Harris and P.C. Parks, Interpolation capabilities of the binary CMAC, *Neural Networks* **6** (3), 429–440 (1993).

Bryson, Jr., A.E. and Y. Ho, *Applied Optimal Control*, Hemisphere Publishing Corporation, Washington, DC, (1975).

Daunicht, W.J., Neural networks mediating linearizable dynamic redundant sensori-motor reflexes characterized by minimum of hermitian norm, In *IEEE International Conference on Neural Networks*, San Diego, CA, July 24–27, 1988, Vol. II.

Franklin, J.A., Refinement of robot motor skills through reinforcement learning, In *Proceedings of the 27th IEEE Conference on Decision and Control*, Austin, TX, December 1988.

Franklin, J.A., Input representation for refinement learning control, In *Proceedings of the 4th International IEEE Symposium on Intelligent Control*, Albany, NY, September 1989.

Franklin, J.A. and O.G. Selfridge, Some new directions for adaptive control theory in robotics, In *Neural Networks for Control*, (Edited by W. Miller, III, R.S. Sutton, P. Werbos), MIT Press, (1990).

Fukuda, T., T. Shibata, M. Tokita and T. Mitsuoka, Adaptation and learning for robotic manipulator by neural network, In *Proceedings of the 29th Conference on Decision and Control*, Honolulu, HI, December 1990, pp. 3283–3288.

Fukuda, T., T. Shibata, M. Tokita and T. Mitsuoka, Neural network application for robotic motion control: Adaptation and learning, In *International Joint Conference on Neural Networks*, San Diego, CA, June 17–19, 1990, Vol. III, pp. 447–452.

Gomi, H. and M. Kawato, Learning control for a closed loop system, In *Proceedings of the 29th Conference on Decision and Control*, Honolulu, HI, December 1990, pp. 3289–3294.

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

Grossberg, S., Adaptive pattern classification and universal coding: II. Feedback, oscillation, olfaction, and illusions, *Biological Cybernetics* **23**, 187–207 (1976).

Guez, A. and I. Bar-Kana, Two degree-of-freedom robot neurocontroller, In *Proceedings of the 29th Conference on Decision and Control*, Honolulu, HI, December 1990, pp. 3260–3264.

Gupta, M.M. and D.H. Rao, Dynamic neural units with applications to the control of unknown nonlinear systems, *Journal of Intelligent and Fuzzy Systems* **1** (1), 73–92 (1993).

Hashimoto, H., T. Kubata, M. Kudou and F. Harashima, Visual control of a robotic manipulator using neural networks, In *Proceedings of the 29th Conference on Decision and Control*, Honolulu, HI, December 1990, pp. 3295–3302.

Hoptroff, R.G., T.J. Hall and R.E. Burge, Experiments with a neural controller, In *International Joint Conference on Neural Networks*, San Diego, CA, June 17–19, 1990, Vol. III, pp. 735–740.

Jang, J.S.R., Self-learning fuzzy controllers based on temporal backpropagation, *IEEE Transactions on Neural Networks* **3** (5), 714–723 (September 1992).

Josin, G., D. Charney and D. White, Robot control using neural networks, In *IEEE International Conference on Neural Networks*, San Diego, CA, July 24–27, 1988, Vol. II.

Kawato, M., Computational schemes and neural network models for formation and control of multijoint arm trajectory, In *Neural Networks for Robotics and Control*, (Edited by W.T. Miller, R. Sutton and P. Werbos), MIT Press, Cambridge, MA, (1990).

le Cun, Y., A theoretical framework for backpropagation, In *Proceedings of 1988 Connectionist Models Summer School*, San Mateo, CA, June 17–26, 1988, (Edited by D. Touretzky, G. Hinton and T. Sejnowski), pp. 21–28.

Martinetz, T.M. and K.J. Schulten, Hierarchical neural net for learning control of a robot's arm and gripper, In *International Joint Conference on Neural Networks*, San Diego, CA, June 17–19, 1990, Vol. III, pp. 747–752.

Miller, W.T., R.P. Hewes, F.H. Glanz and L.G. Kraft, Real time dynamic control of an industrial manipulator using a neural-network-based learning controller, *IEEE Trans. Robotics Automat.* **6**, 1–9 (February 1990).

Nabhan, T.M. and A.Y. Zomaya, Toward generating neural networks structures for function approximation, *Neural Networks* **7** (1), 89–99 (1994).

Nielsen, R.H., *Neurocomputing*, Addison-Wesley, Reading, MA, (1990).

Okuma, S., A. Ishiquro, T. Furuhashi and Y. Uchikawa, A neural network compensator for uncertainties of robotic manipulators, In *Proceedings of the 29th Conference on Decision and Control*, Honolulu, HI, December 1990, pp. 3303–3307.

Patrikar, A. and J. Provence, A self-organizing controller for dynamic processes using neural networks, In *International Joint Conference on Neural Networks*, San Diego, CA, June 17–19, 1990, Vol. III, pp. 359–364.

Pellionisz, A.J., Intelligent decisions and dynamic coordination: Properties of geometrical representation by generalized frames intrinsic to neural and robotic systems, In *IEEE International Conference on Neural Networks*, San Diego, CA, July 24–27, 1988, Vol. II.

Scott, R.W. and D.J. Collins, Neural network adaptive controllers, In *International Joint Conference on Neural Networks*, San Diego, CA, June 17–19, 1990, Vol. III, pp. 381–386.

Snyder, M.M. and D.K. Ferry, Open loop stability criterion for layered and fully-connected neural networks, *Neural Networks* **1** (1), 133 (Suppl 1988).

Sobajic, D.J., J.-J. Lu and Y.-H. Pao, Intelligent control of the Intelledex 605T robot manipulator, In *IEEE International Conference on Neural Networks*, San Diego, CA, July 24–27, 1988 Vol. II.

Suykens, J.A., B.L. De Moor and J. Vandewalle, Static and dynamic stabilizing neural controllers, applicable to transition between equilibrium points, *Neural Networks* **7** (5), 819–831 (1994).

Tsutsumi, K., K. Katayama and H. Matsumoto, Neural computation for controlling the configuration of 2-dimensional truss structure, In *IEEE International Conference on Neural Networks*, San Diego, CA, July 24–27, 1988, Vol. II.

Venugopal, K.P. and S.M. Smith, Improving the dynamic response of neural network controllers using velocity reference feedback, *IEEE Transactions on Neural Networks* **4** (2), 355–357 (March 1993).

Werbos, P., Maximizing long-term gas industry profits in two minutes in Lotus using neural network methods, *IEEE Trans. System, Man, Cybernetics* (March/April 1989).

Werbos, P.J., Consistency of HDP applied to a simple reinforcement learning problem, *Neural Networks* (March 1990).

Williams, R., Adaptive state representation and estimation using recurrent networks, In *Neural Networks for Robotics*, (Edited by W. Miller, III, R.S. Sutton and P. Werbos), MIT Press, (1990).