North-Holland Mathematical Library

VOLUME 51

NORTH-HOLLAND
AMSTERDAM • LONDON • NEW YORK • TOKYO

Mathematical Approaches to Neural Networks

Edited by

J.G. TAYLOR
Centre for Neural Networks
Department of Mathematics
King's College London
London, U.K.

1993
NORTH-HOLLAND
AMSTERDAM • LONDON • NEW YORK • TOKYO

Table of Contents

Mathematical Approaches to Neural Networks
J.G. Taylor (Editor)

Control theory approach

Panos J. Antsaklis
Department of Electrical Engineering, University of Notre Dame, Notre Dame, Indiana 46556, USA

Abstract
The control of complex dynamical systems is a very challenging problem especially when there are significant uncertainties in the plant model and the environment . Neural networks are being used quite successfully in the control of such systems and in this chapter the main approaches are presented and the advantages and drawbacks are discussed. Traditional control methods are based on firm and rigorous mathematical foundations, developed over the last hundred years and so it is very desirable to develop corresponding results when neural networks are used to control dynamical systems.

1. INTRODUCTION

Problems studied in Control Systems theory involve dynamical systems and require real time operation of control algorithms. Typically, the system to be controlled, called *the plant*, is described by a set of differential or difference and perhaps nonlinear equations; the equations for the decision mechanism, the *controller,* are then derived using one of the control design methods. The controller is implemented in hardware or software to generate the appropriate control signals; actuators and sensors are also necessary to translate the control commands into control actions and the values of measured variables into appropriate signals. Examples of control systems are the autopilots in airplanes, the pointing mechanisms of space telecommunication antennas, speed regulators of machines on the factory floor, controllers for emissions control and suspension systems in automobiles, controllers for temperature and humidity regulators at home, to mention but a few. The model of the plant to be controlled can be quite poor either because of lack of knowledge of the process to be controlled, or by choice to reduce the complexity of the control design. *Feedback* is typically used in control systems to deal with uncertainties in the plant and the environment and achieve robustness in stability and performance. If the control goals are demanding, that is the control specifications are tight, while the uncertainties are large then fixed robust controllers may not be adequate. Adaptive control may be used in this case, where the new plant parameters are identified on line and this information is used to change the coefficients of the controller. The area is based on firm mathematical foundations, although in practice engineering skill and intuition are used to make the theoretical methods applicable to real practical systems, as it is the case in many engineering disciplines.

Intelligent Autonomous Control Systems. In recent years it has become quite apparent that in order to achieve high autonomy in control systems, that is to be able to control effectively under significant uncertainties even for example when certain types of failures occur (such as faults in the control surfaces of an aircraft), one needs to implement methods beyond conventional control methods. Decision mechanisms such as planning and expert systems are needed together with learning mechanisms and sophisticated FDI (Failure Diagnosis and Identification) methods. One therefore needs to adopt an interdisciplinary approach involving concepts and methods from areas such as Computer Science, Operations Research in addition to Control Systems and this leads to the area of Intelligent Autonomous Control Systems; see Antsaklis and Passino (1992a) and the references therein. A hierarchical functional intelligent controller architecture, as in Fig. 1, appears to offer advantages; note that in the figure the references to pilot, vehicle and environment etc come from the fact that such functional architecture refers to a high autonomy controller for future space vehicles as described in Antsaklis and Passino (1992b) and Antsaklis, Passino and Wang (1991). A three level architecture in intelligent controllers is quite typical: The lower level is called the Execution level and this is where the numerical algorithms are implemented in hardware or software, that is this is where conventional control systems reside; these are systems characterized by continuous states. The top level is the Management level where symbolic systems reside, which are systems with discrete states. The middle level is the Coordination level where both continuous and discrete state systems may be found. See Antsaklis and Passino (1992b) and the references therein for details.

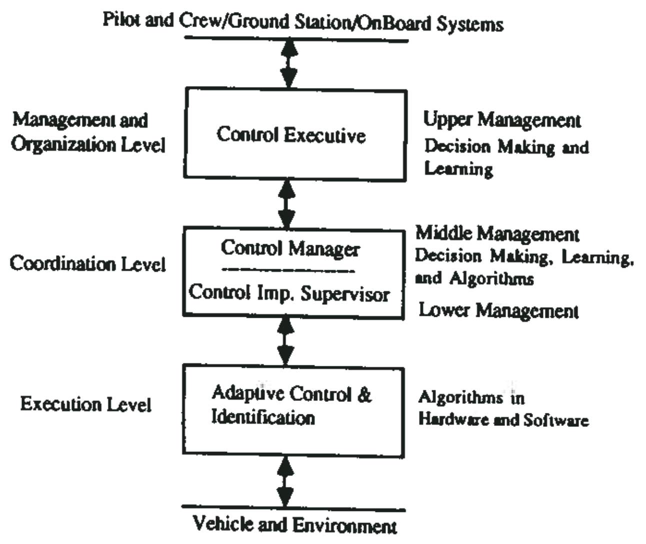
Pilot and Crew/Ground Station/OnBoard Systems
Management and Organization Level
Control Executive
Upper Management
Decision Making and Learning
Coordination Level
Control Manager
Control Imp. Supervisor
Middle Management
Decision Making, Learning, and Algorithms
Lower Management
Execution Level
Adaptive Control & Identification
Algorithms in Hardware and Software
Vehicle and Environment

Figure 1. A hierarchical functional architecture for the intelligent control of high autonomy systems

Neural Networks in Control Systems. At all levels of the intelligent controller architecture there appears to be room for potential applications of neural networks. Note that most of the uses of neural networks in control to date have been in the Execution and Coordination levels - they have been used mostly as plant models and as fixed and adaptive controllers. Below, in the rest of the Introduction a brief summary of the research activities in the area of neural networks in control is given. One should keep in mind that this is a rapidly developing field. Additional information, beyond the scope of this contribution, can be found in Miller, Sutton and Werbos (1990), in Antsaklis (1990), Antsaklis (1992) and in Warwick (1992), which are good starting sources; see also Antsaklis and Sartori (1992).

It is of course well known to the readers that neural networks consist of many interconnected simple processing elements called *units*, which have multiple inputs and a single output. The inputs are weighted and added together. This sum is then passed through a nonlinearity called the *activation function*, such as a sigmoidal function like $f(x) = 1/(1 + e^{-x})$ or $f(x) = \tanh(x)$, or a gaussian-type function, such as $f(x) = \exp(-x^2)$, or even a hard limiter or threshold function, such as $f(x) = \mathrm{sign}(x)$ for $x \neq 0$. The terms artificial neural networks or connectionist models are typically used to describe these processing units and to distinguish them from biological networks of neurons found in living organisms. The processing units or neurons are interconnected, and the strength of the interconnections are denoted by parameters called *weights*. These weights are adjusted, depending on the task at hand, to improve performance. They can be either assigned values via some prescribed off-line algorithm, while remaining fixed during operation, or adjusted via a learning process on-line. Neural networks are classified by their network structure topology, by the type of processing elements used, and by the kind of learning rules implemented.

Several types of neural networks appear to offer promise for use in control systems. These include the multi-layer neural network trained with the back-propagation algorithm commonly attributed to Rumelhart *et al.* (1986), the recurrent neural networks such as the feedback network of Hopfield (1982), the cerebellar model articulation controller (CMAC) model of Albus (1975), the content-addressable memory of Kohonen (1980), and the gaussian node network of Moody and Darken (1989). The choice of which neural network to use and which training procedure to invoke is an important decision and varies depending on the intended application.

The type of neural networks most commonly used in control systems is the *feedforward multilayer neural network*, where no information is fed back during operation. There is however feedback information available during training. Supervised learning methods, where the neural network is trained to learn input/output patterns presented to it, are typicaly used. Most often, versions of the backpropagation algorithm are used to adjust the neural network weights during training. This is generally a slow and very time consuming process as the algorithm usually takes a long time to converge. However other optimization methods such as conjugate directions and quasi-Newton have also been implemented; see Hertz *et al.* (1991), Aleksander and Morton (1990). Most often the individual neuron activation functions are sigmoidal functions, but also signum or radial basis Gaussian functions. Note that in this work the emphasis is on multilayer neural networks. The reader should keep in mind that there are additional systems and control results involving recurrent networks, especially in system parameter identification; one should also mention the work in associative memories, which are useful in the higher levels of intelligent control systems.

One property of multilayer neural networks central to most applications to control is that of function approximation. Such networks can generate input/output maps which can approximate any continuous function with any desired accuracy. One may have to use a large number of neurons, but any desired approximation of a continuous function can be accomplished with a multilayer network with only one hidden layer of neurons or two layers of neurons and weights; if the function has discontinuities, a two hidden layer network may be necessary-see below, Section 2.2. To avoid large numbers of processing units and the corresponding inhibitively large training times, a smaller number of hidden layer neurons is often used and the generalization properties of the neural network are utilized. Note that the number of inputs and outputs in the neural network are determined by the nature of the data presented to the neural network and the type of output desired from the neural network, respectively.

To model the input/output behavior of a dynamical system, the neural network is trained usind input/output data and the weights of the neural network are adjusted most often using the backpropagation algorithm. The objective is to minimize the output error (sum of squares) between the neural network output and the output of the dynamical system (output data) for a specified set of input patterns. Because the typical application involves nonlinear systems, the neural network is trained for particular classes of inputs and initial conditions. The underlying assumption is that the nonlinear static map generated by the neural network can adequately represent the system's behavior in the ranges of interest for the particular application. There is of course the question of how accurately a neural network, which realizes a static map, can represent the input/output behavior of a dynamical system. For this to be possible one must provide to the neural network information about the history of the system-typically delayed inputs and outputs. How much history is needed depends on the desired accuracy. There is a tradeoff between accuracy and computational complexity of training, since the number of inputs used affects the number of weights in the neural network and subsequently the training time. One sometimes starts with as many delayed signals as the order of the system and then modifies the network accordingly; it also appears that using a two hidden layer network-instead of a one hidden layer-has certain computational advantages. The number of neurons in the hidden layer(s) is typically chosen based on empirical criteria and one may iterate over a number of networks to determine a neural network that has a reasonable number of neurons and accomplishes the desired degree of approximation.

When a multilayer neural network is trained as a controller, either an open or closed loop controller, most of the issues are similar to the above. The difference is that the desired output of the neural network, that is the controller generated appropriate control input to the plant, is not readily available, but has to be derived from the known desired plant output. For this, one may use the mathematical model of the plant if available, or some approximation based on certain knowledge of the process to be controlled; or one may use a neural model of the dynamics of the plant or even of the dynamics of the inverse of the plant if such models have been derived. Neural networks may be combined to both identify and control the plant, thus implementing an adaptive controller.

In the above, the desired outputs of the neural networks are either known or they can be derived or approximated. Then, *supervised learning* via the backpropagation algorithm can be used to train the neural networks. Typical control problems which can be solved in this way are problems where a desired output is known. Such is the case in designing a controller to track a desired trajectory; the error then to be minimized is the sum of the squares of the errors between the actual and desired points along the trajectory. There are control problems where no desired trajectory is known but the objective is to minimize say the control energy needed to reach some goal state(s). This is an example of a problem where minimization over time is required and the effect of present actions on future consequences must be used to solve it. Two promising approaches for this type of problems are either constructing a model of the process and then using some type of backpropagation through time procedure, or using an adaptive critic and utilizing methods of reinforcement learning. These are discussed below.

Neural networks can also be used to detect and identify system failures, and to help store information for decision making, thus providing for example the knowledge to decide when to switch to a different controller among a finite number of controllers.

In general there are potential applications of neural networks at all levels of hierarchical intelligent controllers that provide higher degree of autonomy to systems. Neural networks are useful at the lowest Execution level where the conventional control algorithms are implemented via hardware and software, through the Coordination level, to the highest Organization level, where decisions are being made based on possibly uncertain and/or incomplete information. One may point out that at the Execution level, the conventional control level, neural network properties such as the ability for function approximation and the potential for parallel implementation appear to be most relevant. In contrast, at higher levels, abilities such as pattern classification and the ability to store information in a say associative memory appear to be of most interest.

When neural networks are used in the control of systems it is important that results and claims are based on firm analytical foundations. This is especially important when these control systems are to be used in areas where the cost of failure is very high, for example when human life is threatened, as in aircraft, nuclear plants etc. It is also true that without a good theoretical framework it is unlikely that the area will progress very far, as intuitive invention and tricks cannot be counted on to provide good solutions to controlling complex systems under high degree of uncertainty. The analytical heritage of the control field, was in fact pioneered by the use of a differential equation model by J.C.Maxwell to study certain stability problems in Watt's flyball governor in 1868, and this was a case where the theoretical study provided the necessary knowledge to go beyond what the era of Intuitive Invention in control could provide.

In a control system which contains neural networks it is in general hard to guarantee typical control systems properties such as stability. The main reason is the mathematical difficulties associated with the study of nonlinear systems controlled by highly nonlinear neural network controllers-note that the control of linear systems is well understood and neural networks are typically used to control highly nonlinear systems. In view of the mathematical difficulties encountered in the past in the adaptive control of linear systems controlled by linear controllers, it is hardly surprising that the analytical study of nonlinear adaptive control using neural networks is a difficult problem indeed. Some progress has been made in this area and certain important theoretical results have begun to emerge, but clearly the overall area is still at its early stages of development.

In Section 2, the different approaches used in the modeling of dynamical systems are discussed. The function approximation properties of multilayer neural networks are discussed at length, radial basis networks and the Cerebellar Model Articulation Controller (CMAC) are introduced and the modeling of the inverse dynamics of the plant, used in certain control methods is also discussed. In Section 3, the use of neural networks as controllers in problems which can be solved by supervised learning are discussed; such control problems for example would be following a given trajectory while minimizing some output error. In Section 4, control problems which involve minimization over time are of interest; an example would be minimizing the control energy to reach a goal state-there is not known desired trajectory in this case. Methods such as back propagation through time and adaptive critic with reinforcement learning are briefly discussed. Section 5 discusses other uses of neural networks in the failure detection and identification area (FDI), and in higher level control. Sections 6 and 7 contain the concluding remarks and the references respectively.

2. MODELING OF DYNAMICAL SYSTEMS

2.1 Modeling the dynamics of the plant

In this approach, the neural network is trained to model the plant's behavior, as in Fig. 2. The input to the neural network is the same input used by the plant. The desired output of the neural network is the plant's output.

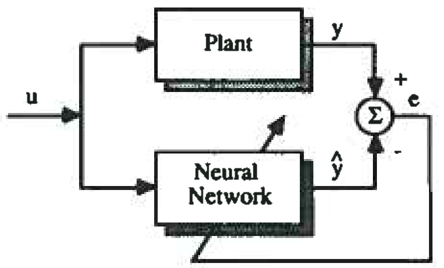

Figure 2. Modeling the plant's dynamics.

The signal $e = y - \hat{y}$ from the summation in Fig. 2 is the error between the

plant's output and the actual output of the neural network. The goal in training the neural network is to minimize this error. The method to accomplish this varies for the type of neural network used and the type of training algorithm chosen. In the figure, the use of the error to aid in the training of the neural network is denoted by the arrow passing through the neural network at an angle. Once the neural network has been successfully trained, it is actually an analytical model of the plant that can be further used to design a controller or to test various control techniques via simulation of this neural plant emulator. This type of approach is discussed in Section 3.

In Fig. 2, the type of plant used is not restricted. The plant could be a very well behaved single-input single-output system, or it could be a nonlinear multi-input multi-output system with coupled equations. The actual plant or a digital computer simulation of the plant could be used. The plant may also operate in continuous or discrete time; although for training the neural network, discrete samples of the plants inputs and outputs are often used. If the plant is time-varying, the neural network clearly needs to be updated on-line and so the typical plant considered is time invariant or if tit is time varying it changes quite slowly. The type of information supplied to the neural network about the plant may vary. For instance, the current input, previous inputs, and previous outputs can be used as inputs to the neural network. This is illustrated in Fig. 3 for a plant operating in discrete time. The boxes with the "Δ" symbol indicate the time delay. The bold lines stress the fact that signals with varying amounts of delay can be used. The plant's states, derivatives of the plant's variables, or other measures can be used as the neural network's inputs. This type of configuration is conducive to training a neural network when the information available about the plant is in the form of an input-output table.

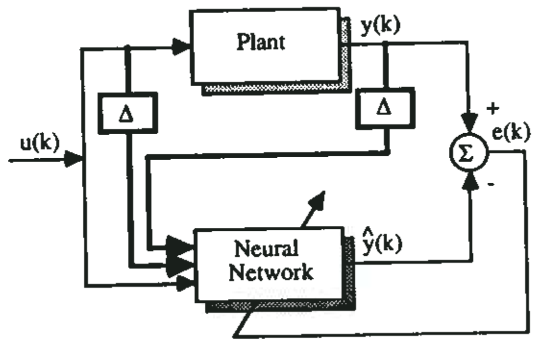

Figure 3. Modeling the discrete time plant's dynamics using delayed signals.

Training a neural network in this manner, by using input-output pairs, ca be viewed as a form of pattern recognition, where the neural network is bein trained to realize some (possibly unknown) relation between two sets. If multi-layer neural network is used to model the plant via the configuratio depicted in Fig. 3, a dynamic system identification can be performed with static model. The past history information needed to be able to model a dynamic systems via a static model is provided by delayed input and output signals. If the back-propagation algorithm is used in conjunction with a multilayer neural network, considerations need to be made concerning which among the current and past values of the inputs and outputs to utilize in training the neural network; this is especially important when the identification is to be on line. In Narendra and Parthasarathy (1990) it is shown that when a series-parallel identification model is used (and the corresponding delayed signals), then the usual backpropagation algorithm can be employed to train the network; when however a parallel identification model is used then a recurrent network results and some type of backpropagation through time, see Section 4, should be used. A moving window of width p time steps could be employed in which only the most recent values are used. An important question to be addressed here concerns the number of delays of previous inputs and outputs to be used as inputs to the neural network; most often the number of delays is taken to be equal to the order of the plant, at least initially.

If there is some apriori knowledge of the plant's operation, this should be incorporated into the training. This knowledge can be imbedded in a linear or nonlinear model of the plant, or incorporated via some other means; see Sartori and Antsaklis (1992a). A possible way of utilizing this information via a plant model is illustrated in Fig. 4; this can be viewed as modeling the unmodelled dynamics of the plant with a neural network.

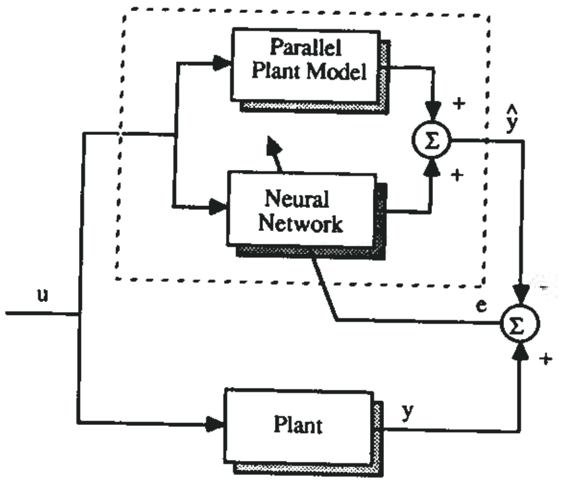

Figure 4. Using apriori knowledge of the plant.

Modeling the plant's behavior via a multilayer sigmoidal neural network has been studied by a number of researchers; see among others Narendra and Parthasarathy (1990), Parthasarathy (1991), Bhat *et al* (1990), Qin, Su and McAvoy (1992), Hou and Antsaklis (1992). In general, the results show that neural networks can be very good models of dynamical systems behavior. This is of course true for stable plants, for certain ranges of inputs and initial conditions and for time invariant or slowly varying systems.

2.2 Function approximation

Neural networks are useful as models of dynamical systems because o their ability to be universal function approximators. In particular, it turns ou that feedforward neural nets can approximate arbitrarily well any continuou function; this in fact can be accomplished using a feedforward network with single hidden layer of neurons with a linear output unit.

More specifically, consider the nonlinear map g: $R^m \rightarrow R$ where

$$g(\mathbf{u}) = \sum_{i=1}^{n} w_i^1 \, S(\mathbf{w}'\mathbf{u}) \tag{1}$$

with $\mathbf{u} \in R^m$ the input vector, $\mathbf{w}' = [w_{i1}, \ldots, w_{im}]$ the input layer weights an w_i^1 the output layer (first layer) weights of unit i. S(.) is the unit activatio function. Such maps are generated by networks with a single hidden layer an a linear output unit. Consider now S(x) to be a sigmoid function, defined as function for which $\lim_{x\rightarrow\infty} S(x) = 1$ and $\lim_{x\rightarrow -\infty} S(x) = 0$. Hornik *et al.* (1989 has shown that when the activation sigmoid function S(x) is non-decreasing the above net can approximate an arbitrary continuous function f(**u**) uniforml on compact sets; in fact it is shown that the set of functions g above is dense i the set of f, the continuous functions mapping elements of compact sets in R^n into the real line. Cybenko (1989) extended this result to continuous activatio sigmoid functions S(x); Jones (1990) showed that the result is still valid whe S(x) is bounded. Typical proofs of this important approximation result ar based on the Stone-Weierstrass theorem and use approximations of th function via trigonometric functions, which in turn are approximated by sum of sigmoidal functions. Similar results have appeared in Funahashi (1989) among others. An important point is that the exact form of the activatio function S(x) is not important in the proof of this result, however it may affec the number of neurons needed for a desired accuracy of approximation of given function f. These results show that typically one increases accuracy b adding more hidden units; one then stops when the desired accuracy has bee achieved. It should be noted that for finite number of hidden units, dependin on the function, significant errors may occur; this reminds us of the Gibb phenomenon in Fourier Series. How many hidden units then one needs in th hidden layer? This is a difficult question to answer and it has attracted muc attention. Several authors have shown (with different degrees of ease an generality) that p-1 neurons in the hidden layer suffice to store p arbitrar patterns in the network; see Nilsson (1965), and Baum (1988), Sartori an Antsaklis (1991) for constructive proofs. The answer to the original questio also depends of course on the kind of generalization achieved by the network also note that certain sets of p patterns may be realizable by fewer than p- hidden neurons. The question of adequate approximation becomes mor complicated in control applications where the functions which must b approximated by the neural network may be functions that generate the contro signals, the range and the shape of which may not be known in advance Because of this, the guidelines to select the appropriate number of hidde neurons are rather empirical at the moment.

The above discussion dealt with continuous functions f, which can be approximated by a neural network with one hidden layer of neurons. When the function under consideration has discontinuities, then two hidden layers may have to be used; Cybenko (1988), Chester (1990), Sontag (1990). In control considerations, Sontag has pointed out that one may need a two hidden layer neural network to stabilize certain plants. It is true of course that a two hidden layer network can approximate any continuous function as well. In addition, experimental evidence tends to show that using a two hidden layer network has advantages over a one layer as it requires shorter training time and overall fewer weights. Because of this a two hidden layer network is many times the network of choice.

There are many other issues relevant to function approximation and control applications, such us issues of network generalization, input representation and preprocessing, optimal network architectures, methods to generate networks, methods of pruning and weight decay. These topics are of great interest to the area of neural networks at large, and a number of these are currently attracting significant research efforts. We are not directly addressing these topics here; the interested reader should consult the vast literature on the subject.

2.3 Radial basis networks

To approximate desired functions, networks involving activation functions other than sigmoids can be used. Consider again a feedforward neural network with one hidden layer and a linear output unit. Assume that the hidden neurons have radial basis functions as activation functions, in which case the neural network implements the nonlinear map $g: R^m \to R$ where

$$g(\mathbf{u}) = \sum_{i=1}^{n} w_i^1 \, G(\|\mathbf{u} - \mathbf{c}_i\|) \tag{2}$$

with $\mathbf{u} \in R^m$ the input vector and w_i^1 the output layer (first layer) weights of unit i. G(.) is a radially symmetric activation function, typically the Gaussian function

$$G(\|\mathbf{u} - \mathbf{c}_i\|) = \exp(-s_i \|\mathbf{u} - \mathbf{c}_i\|^2) \tag{3}$$

where $s_i = 1/\sigma_i^2$. The vectors $\mathbf{c}_i$ $i = 1,\ldots,n$ are the centers of the Gaussian function and if for a particular value of the input $\mathbf{u} = \mathbf{c}_i$ then the ith unit gives an output of +1. The deviation σ_i controls the width of the Gaussian and for large $\|\mathbf{u} - \mathbf{c}_i\|$, more than 3 σ, the output of the neuron is negligible; in this way, practically only inputs in the locality of the center of the Gaussian contribute to the neuron output . It is known from Approximation Theory, see for example Poggio and Girosi (1990), that radially symmetric functions, as $g(\mathbf{u})$ above, with appropriate values for the coefficients in fact minimize the sum of the squares of the errors between $g(\mathbf{c}_i)$ and some desired value subject to certain smoothness constraints. It is then hardly surprising that single hidden layer neural networks with gaussian hidden units and linear output unit, can approximate any continuous fuction f as it was the case when sigmoidals were used. The Stone-Weierstrass theorem can be utilized to prove this result, namely that $g(\mathbf{u})$ above can approximate any continuous function $f: R^m \to R$ on compact sets. The issue now is how to select the weights w_i^1 , centers $\mathbf{c}_i$ and deviations s_i to achieve some desired approximation. For large number of points methods from pattern recognition can be used to determine $\mathbf{c}_i$ and s_i ; in Moody and Darken (1989) k-means clustering algorithms are used for the centers and P nearest neighbor algorithms for the deviations. For small number of points the centers and deviations can be empirically selected to satisfy the problem requirements Parthasarathy (1991); an algorithm is given in Sartori and Antsaklis (1991b) to satisfy certain constraints between given points (see also Sartori and Antsaklis (1992) for similar algorithms but for networks involving sigmoids). After the centers and the deviations have been decided upon, the problem is linear with respect to the output weigths w_i^1 as it is clear from the expression for g above. This can be used to advantage as now linear system identification results may be used to derive provably stable adaptive control systems; this line of inquiry has been pursued successfully in Parthasarathy (1991) and elsewhere.

2.4 CMAC

Albus (1975) introduced the Cerebellar Model Articulation Controller (CMAC), which is briefly described below. CMAC networks have been gaining popularity in control applications especially in robotics, but also in signal, speech and image processing; see Miller, Glanz and Kraft (1990), Kraft and Campagna (1990). The main reasons for the appeal of the CMAC are the speed of training and the fact that it can be inplemented in hardware quite easily using logic cell arrays; note also that CMAC accepts real numbers as input and outputs also real numbers and that it exhibits local generalization properties. However, the size of required memory, which could be in the thousands, and the collisions caused by Hash coding may cause difficulties.

The input to CMAC is quantized and mapped to a number (C) of consecutive *association cells*; C may be for example 32, 256 or larger. There is significant overlap between the association cells excited by different input levels; that is each association cell, also called state-space detector, is excited by a number of input levels. Overlap determines generalization. Each association cell is assigned an address which is then mapped via some Hash coding to a location in a memory where the *weights* are stored. Collisions occur when two different association cells are mapped to the same location. The weights of all active memory locations are then summed to produce the output. That is

$$y^j = \Sigma_{i \text{ active}} \, w_i x_i^j \tag{4}$$

where x_i^j is 1 for i active and 0 otherwise; w_i is the weight in the ith memory location and the superscript j denotes the jth input pattern or level. In CMAC networks, the training algorithms determine the weights; all other parameters, C, overlap, size of memory etc are design parameters fixed in advance. To determine the weights, supervised learning methods, typically the least mean square (LMS) training rule, are used. Each training input u^j produces a vector $\mathbf{x}^j$ of 1s and 0s which indicate the memory locations that have been excited; this is determined by the fixed interconnections. Then the weights are updated by the LMS rule

$$\Delta w = \eta (d_j - \mathbf{w}' \mathbf{x}_j) \mathbf{x}_j \qquad (5)$$

where d_j is the desired output. For convergence results see Wong and Sideris (1992).

CMAC networks are being used in control systems to implement adaptive controllers; the speed of convergence of the training algorithm allows the use of CMAC on line for adaptive control. They are also being used to model dynamical systems, to implement fuzzy controllers etc. One should expect to see more applications of CMAC in the future.

2.5 Models of inverse of the plant

Instead of training a neural network to identify the forward dynamics of the plant as discussed in Section 2.1, a neural network can be trained to identify the inverse dynamics of the plant as illustrated in Fig. 5. This type of model is useful in certain control approaches as discussed in the next section. The neural network's input is the plant's output, and the desired neural network output is the plant's input.

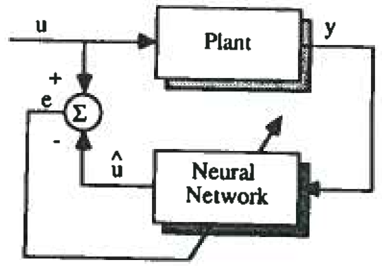

Figure 5. Modeling the plant's inverse dynamics.

The error $e = u - \hat{u}$ is to be minimized and can be used to train the neural network. The type of neural network and the training algorithm used are not restricted. The plant can be continuous or discrete and can also be single-input single-output or multi-input multi-output. The desired output of the neural network is the current input to the plant. The type of information used by the neural network to model the inverse dynamics of the plant may vary. For instance, the neural network's inputs may contain the current and previous outputs and the previous inputs of the discrete time plant, as illustrated for the neural network plant emulator in Fig. 3. In addition, other signals, such as the plant's states, derivatives of the plant's variables, or other measures can also be used as inputs to the neural network. When modeling the inverse dynamics of the plant with a neural network, the assumption is being made, either implicitly or explicitly, that the neural network can approximate well the inverse of the plant. This of course means that the inverse exists and it is unique; if not unique then care should be taken with the ranges of the inputs to the network. It also means that the inverse is stable.

3. CONVENTIONAL CONTROLLERS AND SUPERVISED LEARNING

A neural network can also be used as a conventional controller in both oper loop and closed loop configurations. Because of its ability to adjust its parameters via training, it can in principal implement adaptive control laws For on line implementation the speed of convergence of the training algorithm is of course of great importance. Its use as an open loop controller is firs examined.

Open loop control. There are cases in control where the plant is stable and th uncertainty in the plant parameters and the external disturbances is negligibl and can be ignored. Then open loop control, also called feedforward control may be appropriate and neural networks can then be used in that capacity. I this case one may try to approximate the inverse of the plant, if this is possible aiming to achieve the so called ideal control situation. Note that in the trainin of the neural network to model the inverse dynamics of the plant in th previous section, the purpose often is to use the trained neural network exactl as a feedforward controller for the plant in an open loop configuration. Th desired output of the plant is the input to the neural network controller. Thi method is quite popular among researchers attempting to apply neura networks to the control of robot arms.

Instead of training the neural network to model the inverse dynamics of th plant, as in Fig. 5, the neural network can be trained directly as an open loo controller. This is shown in Fig. 6.

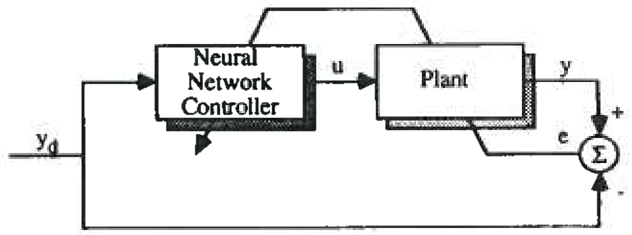

Figure 6. Error back-propagated through plant for an open loop controller.

The error $e = y_d - y$ is used to train the neural network. In thi configuration, there does not exist a desired output for the neural network, an the error information must be "back-propagated" through the plant to accoun for this. The arrow passing through both the plant and the controlle represents the back-propagated error. Note that this situation always arises i both open and closed loop control whenever supervised learning is used; th desired neural network output is not known and must be determined to use sa the backpropagation algorithm. For the backpropagation algorithm the firs derivative of the plant output with respect to the input to the plant is computed This can be approximated by slightly changing the input to the plant an determining the plant's new output. In using this approach to train the neura network, the plant can be thought of as being the "output layer" of the multi layer neural network. Similarly if the mathematical model of the plant i available then these derivatives may be calculated directly. Alternatively, neural network model of the plant can also be used. If a multi-layer neural network is trained to emulate the plant as described in Sect. 2.1, the error can be back-propagated through the neural network model quite easily. If of course the model of the inverse of the plant is available then one can propagate the error forward in the neural network model.

Closed loop control. The use of a neural network in a closed loop configuration as a simple unity or error feedback controller is illustrated in Fig. 7.

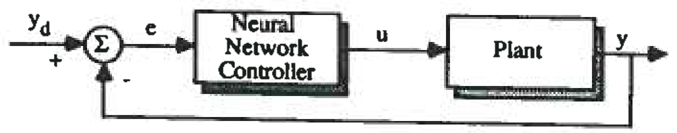

Figure 7. Neural network controller in a conventional closed loop configuration.

Note that the feedback is not restricted to output feedback, state feedback could also be used, and the desired output y_d can be zero as in a regulator loop. As previously discussed, signals, other than e, can also be used as inputs to the neural network. Note that the desired output of the neural network must be determined, from the known desired plant output, before any supervised learning algorithm such as the backpropagation algorithm can be used. Methods to accomplish this were discussed in the open loop control case above. Once trained, the neural network controller can be updated on-line to cope with unforeseen situations or with a slowly time-varying plant. The neural network can also be trained to be a part of an existing control structure; for instance, as part of an internal model control scheme or in conjunction with a PID controller. For this application, the neural network is trained to perform an operation or to augment the operation of an existing controller. The issues involved however are the same as in the case of the simple unity feedback controller shown in Fig. 7.

As another application of a neural network controller, the neural network can be trained to mimic an existing control law. This is illustrated in Fig. 8.

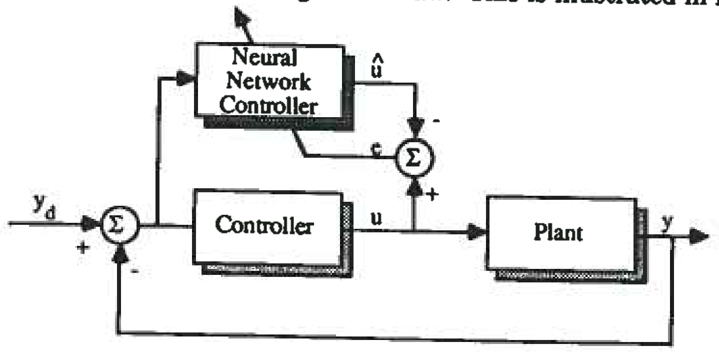

Figure 8. Neural network trained to mimic existing controller.

This use for a neural network is plausible if the controller currently in use is for example too expensive or unreliable. In addition, after the neural network has replaced the current controller, it can be adjusted via training thus taking into account variations in the plant and the environment. For training, the neural network is given the same inputs as the current controller, and the desired neural network output is the output of the current controller. This use of a neural network is equivalent to the design and use of an expert system to capture the reasoning process of an expert. Note that the approach described in Fig. 8 is quite versatile. If a controller is designed to use signals from the plant which are not available or too expensive to measure or compute, the controller can not be directly implemented; however, if a neural network can be trained to emulate the already designed controller using different signals, the neural network can then be used as the actual controller for the plant.

It is also possible to use a neural controller in parallel with an already existing controller to enhance its operation as it was mentioned above, when the control action has deteriorated because of aging or damage to the plant.

Model reference control. Besides training a neural network to mimic a control law, the neural network controller can also be trained to reduce the error at the output of the plant compared to a reference model as illustrated in Fig. 9.

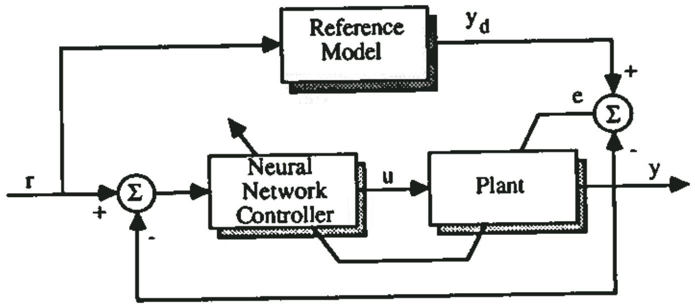

Figure 9. Model reference control. Error propagated through plant.

Here again the desired neural controller output must be determined in order to use the backpropagation algorithm; that is the error must be back-propagated through the plant and then used to adjust the weights of the neural network controller. As discussed previously, the gradient of the error can be approximated by varying the plant's inputs and measuring the resulting outputs or be calculated from the mathematical model of then plant if available. Alternatively, if the plant is first modelled with a multi-layer neural network as discussed in Sect. 2.1, this neural network can be used to replace the plant for the training of the neural network controller, and the back-propagation of the error through the plant emulator can then be accomplished; similar method is used if a neural model of the inverse of the plant is available.

It is clear that neural networks can be used in model reference control configurations other than the one shown in Fig. 9. The issues are the same and the difficulties and problems encountered are similar. As a side comment, by using a sigmoidal nonlinearity at the output of the neural network, the saturation property of most actuators that follow the controller can be accommodated, since the range of the sigmoid function is from -1 to +1 (tanh(x)) and can be appropriately shifted and scaled.

There are many references of successful neural controllers; see for example Antsaklis (1990 and 1992), Chen (1990), Hou and Antsaklis (1992), Kana and Guez (1989), Kraft and Campagna (1990), Miller, Sutton and Werbos (1990), Narendra and Parthasarathy (1990), Ungar, Powell and Kamens (1990), Ydstie (1990).

4. CONTROL OBJECTIVES OVER TIME AND REINFORCEMENT LEARNING

In all the control problems considered in the previous section a desired trajectory of the plant output was known. This meant that the desired outputs of the neural networks were either known or they could be derived or approximated, and supervised learning via the backpropagation algorithm could be used to train the neural networks. Typical control problems of this type are the regulation and the tracking problems where the plant output must follow a given trajectory. When the desired plant trajectories are not known then supervised learning methods cannot be used to train the neural networks. This is the case for example when the control objective is to minimize the control energy needed to reach some goal state(s). This is an example of a problem where *minimization over time* is required and the effect of present actions on future consequences must be determined to solve it. Two approaches may be used to address this type of problems: either constructing a model of the process and then using some type of *backpropagation through time* (BTT) procedure, or using an *adaptive critic* and utilizing methods of reinforcement learning.

Backpropagation through time (BTT). The basic backpropagation algorithm is an efficient way to calculate the derivatives of the (output) error with respect to a large number of input variables in a feedforward network. BTT extends this method to recurrent networks; see Werbos (1990). BTT is in effect a computationally more efficient first-order calculus of variations. This method uses noise free models to calculate the derivatives of future utility with respect to present actions. BTT can be derived from the basic backpropagation algorithm by unfolding an arbitrary recurrent net into a multilayer feedforward net that grows by one layer at each time step; the algorithm takes its name from the fact that the computation of the error derivatives is based on information propagating from later times to earlier times in the recurrent network. This method is difficult to use in its general form, however specific successful applications have been reported; see for example the robot arm applications of Kawato (1990) and the truck backer-upper of Nguyen and Widrow (1990).

Adaptive critic. It consists of two networks, the action network which is the neural network controller of Fig. 10 below, and the critic network which may or may not be a neural network. It approximates the dynamic programming solution to the problem and it performs well in noisy environment and with inexact models; see Barto (1990). The critic guides how the controller is

adapted; it generates a performance index J and the controller is rewarded when the control u leads to larger J and punished when u leads to smaller J in the next time step. The neural controller needs the gradient of the critic's evaluation J with respect to the control signals. One could use a neural network model of the plant together with the critic for the derivatives. If however only values of performance J but not derivatives are available, as it is the case when a model of the evaluation process is not available, the derivatives must be estimated. In this case *reinforcement learning* could be used to advantage. In reinforcement learning, contrary to supervised learning, there is no knowledge of the desired output and the learning system receives only a measure of performance; see Barto (1990) and Sutton, Barto, and Williams (1992). Note that when supervised learning can be used, it is much more efficient than reinforcement learning.

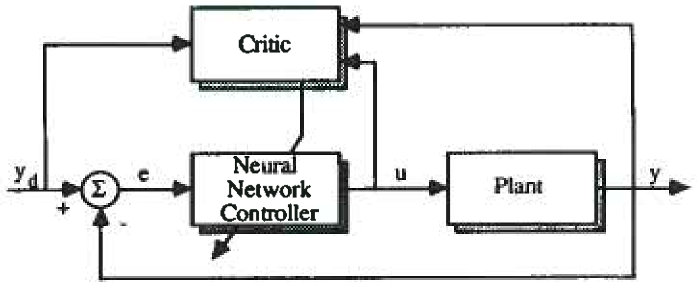

Figure 10. Adaptive critic used to adjust neural network controller.

To design the critic one must be able to determine the current performance so to reward or punish the controller, based on properties of future plant behavior. Optimal control requires accurate plant model and large amounts of computation or mathematical tractability of the model and of the objective function to obtain an analytical solution and this is not usually possible. In the 70s the problem of subgoal performance measures was studied, while in the AI literature progress in the problem of credit-assignment for reinforcement learning has been reported; these are problems related to the design of the critic network and along similar lines of inquiry progress has been made-see Barto (1990) for a comprehensive discussion.

One could of course develop a model of the plant and then use conventional dynamic programmic to solve these minimization over time control problems or one could develop a space/time model and use BTT. These model based methods however are not necessarily effective with inaccurate models and noisy environments. Most often one needs some exploratory search in control space, search which one typically associates with reinforcement learning methods. So the adaptive critic and reinforcement learning methods show great promise; additional progress needs to be made before it is clearly demonstrated that this is the best way to address these optimization over time control problems.

5. FDI, AND HIGHER LEVEL DECISION MAKING

Neural networks discussed in this section are used in the control of the plant, but are not actually in the conventional control loop, per se. As described in Antsaklis and Passino (1992b), see also the Introduction above, they perform some higher level decision making tasks in a manner which adds more autonomy to the system. This configuration is illustrated in Fig. 11. The neural network becomes a high level decision maker and is not directly involved in determining the input to the plant. Instead, the neural network supplies to the controller information to properly form control signals for the plant. This may require adapting the controller based on the information provided by the neural network and in that sense there are similarities between this method and the adaptive critic discussed above when the critic is implemented as a neural network. In Fig. 11, the neural network's inputs are the desired output of the plant, and the actual input and output of the plant. As previously discussed, this can be extended to include other signals as well. This configuration also does not preclude the use of a reference model. The output of the neural network is a signal that is useful for the control of the plant.

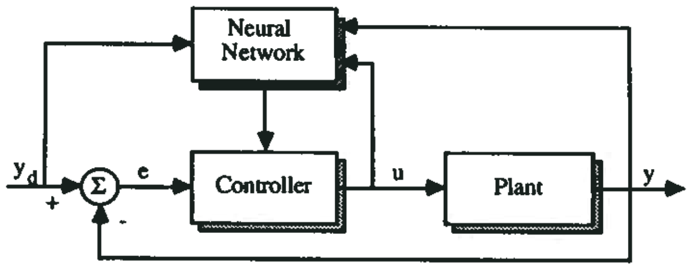

Figure 11. Neural network used as a high level decision maker.

Two uses are discussed here: changing the control parameters and supplying failure information. In the first, the neural network is used to determine appropriate values for parameters in the controller. For example, for a PID controller, the neural network can be trained to determine values for the gains based on the operating conditions of the plant, thus providing parameter tuning. The neural network could also be used as a scheduler; see Sartori and Antsaklis (1991b). Given the current operating point of the plant, the neural network decides which control law to use. Depending on the choice of the neural network implementation, the neural network scheduler may give rise to quite smooth control law switching. As another option, the neural network can be used as an optimizer to find the minimum of a cost function, such as in a llinear quadratic optimal controller. The output of the neural network is the value of the controller parameters that minimize that cost function. This would be an alternative to solving a Ricatti equation at each time step.

Besides being used to determine parameters for the controller, the neural network can also be trained to supply failure information to the controller. Depending on the type of training data used, the type of information can vary from fault detection to fault identification to fault diagnosis. The controller can then use this knowledge to take appropriate actions. A neural network trained for fault detection and identification can even be used in conjunction with another neural network trained to choose the appropriate control parameters given specific failures of the system. There are several references which address FDI problems. See for example Naidu, Zafiriou and McAvoy (1990), Sartori, and Antsaklis (1992b), Yao and Zafiriou (1990) and the references therein.

6. CONCLUSION

The ever-increasing demands of the complex control systems being built today and planned for the future dictate the use of novel and more powerful methods in control. The potential for neural networks in solving many of the problems involved is great, and this research area is evolving rapidly. The viewpoint is taken that conventional control theory should be augmented with neural networks in order to enhance the performance of the control systems. In the tradition of the Systems and Control area, such developments in neural networks for control should be based on firm theoretical foundations and this is still at its early stages. Strong theoretical results guaranteeing control system properties such as stability are still to come, although promising results reporting progress in special cases have been reported recently. The potential of neural networks in control systems clearly needs to be further explored and both theory and applications need to be further developed.

Acknowledgement
This work was completed while being on sabbatical leave of absence at Imperial College in London, England and at the National Technical University of Athens, Greece and I would like to thank these fine Universities for their hospitality.

7. REFERENCES

Albus J.S. (1975), "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)," *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control*, vol. 97, series G, no. 3, pp. 220-227, Sept. 1975.

Aleksander I., Morton H.B. (1990), *An Introduction to Neural Computing*, Chapman and Hall, UK, 1990.

Anderson C. (1990), "Learning to Control an Inverted Pendulum Using Neural Networks," *IEEE Control Systems Magazine*, vol. 10, no. 3, pp. 31-37, April 1990.

Antsaklis P.J. (1990), Editor, Special Issue on "Neural Networks in Control Systems," *IEEE Control Systems Magazine*, vol. 10, no. 3, pp. 3-87, April 1990.

Antsaklis P.J. (1992), Editor, Special Issue on "Neural Networks in Control Systems," *IEEE Control Systems Magazine*, vol. 12, no. 3, April 1992.

Antsaklis P.J., Passino K.M. (1992a), Editors, *Introduction to Intelligent and Autonomous Control*, Kluwer, 1992.

Antsaklis P.J., Passino K.M. (1992b), "Introduction to Intelligent Control Systems with High Degree of Autonomy", *Introduction to Intelligent and Autonomous Control*, P.J.Antsaklis and K.M.Passino, Eds., Chapter 1, Kluwer, 1992.

Antsaklis P.J., Passino K.M., Wang S.J. (1991), "An Introduction to Autonomous Control Systems," *IEEE Control Systems Magazine*, vol. 11, no. 4, pp. 5-13, June 1991.

Antsaklis P.J., Sartori M.A. (1992), "Neural Networks in Control Systems", *Encyclopedia of Systems and Control, Supplementary Volume II*, M.G. Singh Editor-in-chief. To appear.

Barto A.G. (1990), "Connectionist Learning for Control: An Overview," *Neural Networks for Control*, Miller, Sutton and Werbos, Editors, MIT Press, 1990.

Baum E.B. (1988), "On the Capabilities of Multilayer Perceptrons," *J. Complexity*, vol 4, pp 193-215, 1988.

Bhat N.V., Minderman P., McAvoy, Wang N. (1990), "Modeling Chemical Process Systems via Neural Computation," *IEEE Control Systems Magazine*, vol. 10, no. 3, April 1990.

Chen F-C. (1990), "Back-Propagation Neural Networks for Nonlinear Self-Tuning Adaptive Control," *IEEE Control Systems Magazine*, vol. 10, no. 3, pp. 44-48, April 1990.

Chester D. (1990), " Why Two Hidden Layers are Better than One," *Proc Int. Joint Conf. on Neural Networks*, IEEE Publications, ppI.265-268, 1990.

Cybenko G. (1988), "Continuous Valued Neural Networks with Two Hidden Layers are Sufficient," Tech Report, Dept of Computer Science, Tufts University, 1988.

Cybenko G. (1989), "Approximations by Superpositions of a Sigmoidal Function," *Math of Control, Signals, and Systems*, vol 2, no 4, pp 303-314, 1989.

Funahashi K.(1989), "On the Approximate Realization of Continuous Mappings by Neural Networks," *Neural Networks*, vol 2, pp183-192, 1989; also in *Proc Int. Joint Conf. on Neural Networks*, IEEE Publications, ppI.641-648, 1988.

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

Hertz J., Krogh A., Palmer R.G. (1991), *Introduction to the Theory of Neural Computation*, Addison-Wesley.

Hopfield J.J. (1982), "Neural Networks and Physical Systems with Emergent Collective Computational abilities," *Proceedings of the National Academy of Science, U.S.A.*, vol. 79, pp. 2554-2558, April 1982.

Hornik K.M., Stinchocombe M., White H. (1989), "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, vol. 2, pp. 359-366, 1989.

Hoskins J.C., Himmelblau D.M. (1988), "Artificial Neural Network Models of Knowledge Representation in Chemical Engineering," *Computers and Chemical Engineering*, vol. 12, pp. 881-890, 1988.

Hou Z., Antsaklis P.J. (1992), *Analysis of Auto Powertrain Dynamics and Modeling using Neural Networks*, MSc Thesis, Department of Electrical Engineering, University of Notre Dame, May 1992.

Jones L.K. (1990), "Constuctive Approximations for Neural Networks by Sigmoidal Functions," *IEEE Proceedings*, vol 78, no 10, pp 1586-1589, 1990.

Kana I., Guez A. (1989), "Neuromorphic Adaptive Control," Proc of the 28th IEEE Conf on Decision and Control, pp. 1739-1743, 1989.

Kawato M. (1990), "Computational Schemes and Neural Network Models for Formation and Control of Multijoint Arm Trajectory," *Neural Networks for Control*, Miller, Sutton and Werbos, Editors, MIT Press, 1990.

Kohonen T.(1980), *Content-Addressable Memories*, Springer-Verlag, New York, NY, 1980.

Kraft L.G., Campagna D.P. (1990), "A Comparison Between CMAC Neural Network Control and Traditional Adaptive Control Systems," *IEEE Control Systems Magazine*, vol. 10, no. 3, pp. 36-43, April 1990.

Leonard J.A., Kramer M.A. (1990), "Classifying Process Behavior with Neural Networks: Strategies for Improved Training and Generalization," *Proceedings of the 1990 American Control Conference*, pp. 2478-2483, 1990.

Lippmann R.P.(1987), "An Introduction to Computing with Neural Networks,' *IEEE ASSP Magazine*, pp. 4-22, April 1987.

Miller T.W., Glanz F.H., Kraft L.G. (1990), "CMAC: An Associative Neural Network Alternative to Backpropagation," *IEEE Proceedings*, vol 78, no 10 pp 1561-1567, 1990.

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

Moody J., Darken D.J. (1989), "Fast Learning in Networks of Locally-Tuned Processing Units," *Neural Computation*, vol. 1, pp. 281-294, 1989.

Naidu S.R., Zafiriou E., McAvoy T.J.(1990), "Use of Neural Networks for Sensor Failure Detection in a Control System," *IEEE Control Systems Magazine*, vol. 10, pp. 49-55, April 1990.

Narendra K.S., Parthasarathy K.(1990), "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Transactions on Neural Networks*, vol. 1, no 1, pp. 4-27, March 1990.

Nguyen H.D., Widrow B. (1990), "Neural Networks for Self-Learning Control Systems," *IEEE Control Systems Magazine*, vol. 10, no. 3, pp. 18-23, April 1990.

Nilsson N.J. (1965), *Learning Machines*, McGraw-Hill, 1965.

Parthasarathy K.(1991), *Identification and Control Using Neural Networks*, PhD dissertation, Dept of Elec Engr, Yale University, Dec 1991.

Passino K.M., Sartori M.A., Antsaklis P.J. (1989), "Neural Computing for Numeric-to-Symbolic Conversion in Control Systems," *IEEE Control Systems Magazine*, pp. 44-52, April 1989.

Poggio T., Girosi F. (1990), "Networks for Approximation and Learning," *IEEE Proceedings*, vol 78, no 9, pp 1481-1497, 1990.

Psaltis D., Sideris A., Yamamura A. (1987), "Neural Controllers", *IEEE 1st Intl Conf on Neural Networks*, vol 4, pp 551-558, 1987.

Qin S-Z, Su H-T, McAvoy T.J. (1992), "Comparison of Four Neural Net Learning Methods for Dynamic System Identification," *IEEE Trans on Neural Networks*, vol 3, no 1, pp 122-130, 1992.

Rumelhart D.E., Hinton G.E., Williams R.J. (1986), "Learning Internal Representations by Error Propagation," in Rumelhart D.E., McClelland J.L., eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol 1: Foundation*, pp. 318-362, MIT Press, 1986.

Sartori M.A. (1991), *Feedforward Neural Networks and their Application in the Higher Level Control of Systems*, Ph.D. Dissertation, Department of Electrical Engineering, University of Notre Dame, April 1991.

Sartori M.A., Antsaklis P.J. (1990), "Neural Network Training via Quadratic Optimization," Technical Report # 90-05-01, Department of Electrical Engineering, University of Notre Dame, May 1990, Revised April 1991. Also in *Proc of ISCAS*, San Diego, CA, May 10-13, 1992.

Sartori M.A., Antsaklis P.J. (1991a), "A Simple Method to Derive Bounds on the Size and to Train Multilayer Neural Networks," *IEEE Trans on Neural Networks*, vol 2, no 4, pp 467-471, July 1991.

Sartori M.A., Antsaklis P.J. (1991b), "A Gaussian Neural Network Implementation for Control Scheduling," *Proceedings of the 1991 IEEE International Symposium on Intelligent Control*, pp. 400-404, August 1991.

Sartori M.A., Antsaklis P.J. (1992a), "Implementations of Learning Control Systems Using Neural Networks," *IEEE Control Systems Magazine*, in Special Issue on 'Neural Networks in Control Systems', Vol.12, No.3, April 1992.

Sartori M.A., Antsaklis P.J. (1992b), "Failure Behavior Identification for a Space Antenna via Neural Networks", *Proc of American Control Conference*, Chicago, IL, June 24-26, 1992.

Sartori M.A., Passino K.M., Antsaklis P.J. (1989), "Artificial Neural Networks in the Match Phase of Rule-Based Expert Systems," *Proceedings of the Twenty-Seventh Annual Allerton Conference on Communication, Control, and Computing*, University of Illinois at Urbana-Champaign, pp. 1037-1046, September 27-29, 1989.

Sartori M.A., Passino K.M., Antsaklis P.J.(1992), "Neural Networks in the Match Phase of Rule-Based Expert Systems," *IEEE Transactions on Knowledge & Data Engineering*. To appear.

Sontag E.D. (1990), "Feedback Stabilization using Two Hidden Layer Nets," Report SYCON-90-11, Rutgers Center for Systems and Control, Rutgers University, 1990.

Sontag E.D. (1991), "Feedforward Nets for Interpolation and Classification," Report SYCON-, Rutgers Center for Systems and Control, Rutgers University, 1991.

Sutton R.S., Barto A.G., Williams R.J. (1992), "Reinforcement Learning is Direct Adaptive Optimal Control," *IEEE Control Systems Magazine*, vol. 12, no. 3, April 1992.

Yao S.C., Zafiriou E. (1990), "Control System Sensor Failure Detection via Networks of Localized Receptive Fields," *Proceedings of the 1990 American Control Conference*, pp. 2472-2477, 1990.

Ydstie B.E. (1990), "Forecasting and Control Using Adaptive Connectionist Networks," *Computers in Chemical Engineering*, vol 14, pp. 583-599, 1990.

Ungar L.H., Powell B.A., Kamens S. (1990), "Adaptive Networks for Fault Diagnosis and Process Control," *Computers in Chemical Engineering*, 1990.

Warwick K. (1992), Editor, *Neural Networks for Control and Systems: Principles and Applications*, Peter Peregrinus, UK, 1992.

Werbos P.J. (1990), "Backpropagation Through Time: What it Does and How to Do it," *IEEE Proceedings*, vol 78, no 10, pp 1550-1560, 1990.

Wong Y., Sideris A. (1992), "Learning Convergence in the Cerebellar Model Articulation Controller," *IEEE Trans on Neural Networks*, vol 3, no 1, pp 115-121, Jan 1992.