NEURAL NETWORK TECHNIQUES FOR STABLE LEARNING CONTROL OF NONLINEAR SYSTEMS

Approved by
Dissertation Committee:

Dedicated to Austin and Nathan

NEURAL NETWORK TECHNIQUES FOR STABLE LEARNING CONTROL OF NONLINEAR SYSTEMS

by

PHILIP DALE HEERMANN, B.S., M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 1992

ACKNOWLEDGEMENTS

First, I would like to thank Billy V. Koen, who as my principal advisor provided guidance and encouragement toward the completion of this dissertation. Appreciation is also due to A.T. Campbell, Hal FitzPatrick, James Keeler, and Eric Hartman for the many invaluable discussions of neural networks, controls and nonlinear systems. Thanks also to ALFA Engineering Inc. for providing additional computational resources.

Finally, special thanks to my wife, Mae, for encouragement and patience through all of my post-graduate work.

NEURAL NETWORK TECHNIQUES FOR STABLE LEARNING CONTROL OF NON-LINEAR SYSTEMS

Publication No. ______________

Philip Dale Heermann, Ph.D.
The University of Texas at Austin, 1992

Supervisor: Billy V. Koen

This dissertation presents techniques for stable adaptive control of inherently unstable nonlinear systems. The research focuses on techniques that are practical for actual application. The dissertation explores model free artificial neural network control, assuring plant stability, and adaptive controller training. Example tasks are the unstable nonlinear tasks of balancing an upright pole on a cart and balancing an underpowered pendulum. The principal results include nonlinear control via direct attractor construction, a distal-error neural network training algorithm for learning sequences of actions, and the demonstration of artificial neural network learning with active Lyapunov stabilization.

The direct attractor construction is a technique for learning directly to control a nonlinear system without modelling or estimating parameters of the

system. An artificial neural network controller learns to control the system such that a fixed point attractor is produced at the desired setpoint. The technique is fully nonlinear and is not restricted to a particular class of systems.

The second result, a distal-error training algorithm, is a method for training an artificial neural network over a sequence of actions. To develop the algorithm, new mathematical structures and operators are introduced and the necessary algebra and theorems are proven. The training algorithm incorporates features from both supervised learning and reinforcement learning algorithms.

The third principal result is the demonstration of learning while a system is actively Lyapunov bound. This demonstrates that the stability of an unstable system can be assured while training an adaptive controller. This is shown to be necessary to assure the failure free operation of artificial neural network controllers on certain physical systems.

Overall, this dissertation presents artificial neural network control techniques that are practical for use in physical systems. No assumptions were made that would be violated in application of the controller to physical systems.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

1.1 Introduction

Central to the operation of modern technology is system regulation and control. Systems from home appliances and automobiles to nuclear reactors and spacecraft require automatic control systems to maintain their operation. All control systems harness a flux of energy to perform a task. A control system may be required to maintain a stable condition, produce a cyclic action, or follow a specified path. For example, a thermostat controls the heat flow of a furnace to maintain a comfortable living environment, and an automobile engine's cams and ignition system produce a cyclic rotation of the crank shaft.

Most controllers of today are simple in design and are fixed at the time of development. They suffer from limits on the range of operation, instability, improper operation, inefficiency and a multitude of other problems due to their simple design and lack of adaptability. Automatic control researchers are well aware of these problems and have developed specialized algorithms and techniques to incorporate more knowledge into control systems. The work, however, is narrow in focus and according to Karl Astrom (Astrom, 1988) the work has concentrated on specific solutions to precisely defined problems. Because of the narrow focus and the difficulty in applying the specialized techniques, many control tasks continue to be solved using very simple controllers. The process control industry is a good example. Although significant advances in hardware and control theory have occurred, most systems are still implementing PID (proportional-integral-derivative) controllers. These controls use 1930s control theory running on advanced digital hardware.

The main problem with many new techniques is that they require significant knowledge to be properly implemented. What is needed are more general techniques, which are easier and more cost effective to use, but provide improved control over the simple controllers (Astrom, 1988). To obtain improved control, more plant knowledge needs to be incorporated into the controller (Astrom, 1988). This can be accomplished by building controllers that initially contain sufficient knowledge such as expert systems or optimal control. Another approach is to design a controller to adapt or learn the system through time. The second or adaptive method is desirable, since complete knowledge for all time is not generally available *a priori* (Kumar and Guez, 1991).

1.2 Objectives

Research on adaptive controllers was vigorous in the 1950s. Interest in adaptive controllers diminished, however, due to a flight test disaster caused by an early adaptive flight control system (Astrom,1988). The primary difficulty is that the controller may adapt itself improperly and cause a failure rather than improved action. The stability or failure prevention of the system under adaptive or any other automatic control is important. To illustrate the importance of stability, Kuo (Kuo, 1982) distinguishes systems as *useful* if stable and *useless* if unstable.

This dissertation presents a general controller architecture, which addresses the problems with adaptive control by using initial design knowledge and learning from the process through time. The resulting controller automatically acquires system response knowledge to improve its performance while ensuring stability and safety of the system at all times. The control architecture is general in nature and provides self adaptive nonlinear control without linearization of any system component.

The focus of this research is to develop practical techniques for stable nonlinear adaptive control. The goal is to develop techniques to ensure system stability while providing adaptive control of nonlinear systems without fitting a system model. To meet this goal, three objectives for control of nonlinear systems are proposed.

The first objective is to develop a direct means for controlling any nonlinear system without plant modelling. The central idea is to avoid the many problems with fitting models to nonlinear systems by direct attractor construction. Direct attractor construction is a technique for learning to control a nonlinear system directly. An artificial neural network controller learns to control the system such that a fixed point attractor is produced at the desired setpoint. The technique is fully nonlinear and is not restricted to a particular class of systems.

The second objective is to develop a technique for training a neural network controller that receives training information infrequently instead of continuously. This is necessary for training a controller that must perform a series of steps before a teaching signal is available. Therefore, the second objective is to develop a distal-error training algorithm. Derivation of the algorithm requires the development of new mathematical structures and operators. The necessary algebra is demonstrated and theorems are proven.

The training algorithm incorporates features from both supervised learning and reinforcement learning algorithms.

The third primary objective is the demonstration of learning while a system is actively stabilized. One controller assures stability of the system, thus forming an active Lyapunov bound of the system. Simultaneously, a neural network learns to control the system to meet the control objective. The Lyapunov bound is shown to be necessary to assure the failure free operation of artificial neural network controllers on certain physical systems.

These objectives demonstrate that a reflexive stabilizer and an artificial neural network can ensure system stability, while providing adaptive control of difficult nonlinear systems without fitting a system model. The artificial neural network control techniques are practical for use in real systems. No assumptions are made that would be violated in the actual application of the controller.

1.3 Need for Improved Controls

Current automatic controls provide sufficient solutions to today's problems, but they are not agile enough to address many control demands of future systems. The demands for systems include: 1) increased flexibility and autonomy (Erickson, 1990; Peek and Antsaklis, 1990), 2) reduced cost of operation in terms of energy, resources, and the environment (Werbos, 1990b), 3) decreased maintenance (Erickson, 1990), 4) wider range of operation (Yuh, 1990; Sifer *et al*., 1991), and 5) robust operation for complex noisy nonlinear systems (Erickson, 1990; Werbos, 1990b).

Flexible and autonomous operation is needed for systems where all situations cannot be predicted. The main need is for the systems to operate with little or no human guidance. For example, a space probe entering the orbit of a planet may encounter a new moon that it needs to accommodate. A high degree of autonomy is important because a human controller is not immediately present.

Improved efficiency of operation is important for current and future systems. The undesirable side effects of almost any system can be mitigated though improved control. The fuel injection system of a modern automobile is a good illustration. Increased power output and decreased emissions are the results of closely controlling the fuel flow.

Good control also can reduce maintenance of both the system and of the controller. A good controller more tightly and correctly governs the controlled system, thus avoiding undesirable conditions like overheating or other system stresses. Maintenance of controls also can be reduced if the controller can accommodate a wider range of operation and incorporate changes in the system. These improvements result in less re-tuning of the controller.

A wider range of operation is one function required from future controllers. Currently many controllers are designed by linearizing about a given operating condition. The linearization results in severe problems when the operation is moved to a new region. The Space Shuttle's flight controller is a prime example of a controller with limited range. Launches are often delayed until the environmental conditions, like the winds aloft, are within the flight controller's ranges.

The National Aerospace Plane (NAP) (Werbos, 1990b) and other high performance vehicles illustrate the need for control of highly nonlinear systems under very noisy conditions. Improvement in control is a key need of the NAP where human-like adaptive and analysis capabilities are needed at rates ten times faster than humans have handled in the past (Werbos, 1990b). Therefore, better controllers are needed to improve the quality of current systems and enable the systems of the future.

1.4 Scope of Research

This research considers the control of highly nonlinear physical systems with unknown parameters. Inherently unstable systems are considered. Evaluation problems are taken from the domain of dynamic system control.

The control architecture is evaluated by computer simulation of both the evaluation problem and the control architecture. All the studies are conducted under the assumption that basic knowledge is known but specific details are not available. For example, the range and general reaction of the system is known, but specifics like friction in bearings or disturbance inputs are unknown. The control architecture is demonstrated using two computer simulated systems; a low power inverted pendulum and classical cart/pole balancing.

1.5 Overview by Chapters

Chapter II is an overview of dynamic systems and control. Topics discussed include properties of linear and nonlinear systems, important results from recent nonlinear system analysis, and classical and modern control techniques. Artificial neural network foundations, properties and application issues are the focus of Chapter III. Chapter IV summarizes previous studies of neural network controllers and discusses their importance to adaptive control. In Chapter V, the design, motivation and mathematical derivation of the new control architecture is presented. Chapter VI describes the experimental methodology such as the test platform, systems of study, and other practical issues. The results of the simulation studies are the topic of Chapter VII. The final chapter, Chapter VIII, contains the conclusions and final recommendations.

CHAPTER II

DYNAMIC SYSTEMS AND CONTROL

2.1 Dynamic Systems

A dynamic system refers to a system in motion. The study of dynamic systems is the investigation of the variations in systems over time. This dissertation concentrates on physical systems, emphasizing the exchange of mass and energy instead of information flow, biological populations, or pure mathematics.

The language of dynamic systems is the mathematics of differential equations. Thus, the foundations of dynamic system analysis are the generation and analysis of differential equations.

For control applications, the distinction between linear and nonlinear systems is important. According to Wylie and Barrett (1982) a linear system is defined as:

DEFINITION: A differential equation is linear if and only if each term of the equation is of the first degree in its dependent variables or any of its dependent variable derivatives.

In comparison, a nonlinear differential equation is simply any differential equation, which is not linear in the set of all its dependent variables. Real physical systems in general are nonlinear. Thus, they are a motivation for the study of nonlinear systems.

Most work in the domain of automatic control has concentrated on linear systems, mainly for lack of good tools for analysis of nonlinear systems. This is not as great a drawback as it first appears, considering that any nonlinearity can be represented as a truncated Taylor series. As long as the system's operation remains sufficiently close to the state about which equations were linearized, the linear approximation is adequate. Nevertheless, a fully nonlinear representation is desirable, and with the wide availability of digital computers this is feasible.

This chapter describes nonlinear systems and the control of physical systems. It provides a foundation in the fundamental concepts, which are necessary to understand this dissertation. A reader not familiar with controls or dynamic systems may wish to consult the glossary before continuing.

Nonlinear systems are described in Section 2.2. Topics discussed include differences between linear and nonlinear systems, bifurcations, attractors, and an introduction to the phenomena of chaos. Section 2.3 discusses dynamic system stability, presenting the definition of Lyapunov stability and the importance of stability to dynamic systems. Dynamic systems that structurally vary through time are discussed in Section 2.4. Section 2.5 presents recent results in nonlinear systems theory to prove that adaptive control of nonlinear systems is theoretically possible. The fundamentals of physical systems control are described in Section 2.6. The last two sections, 2.7 and 2.8, present common methods of physical system control.

2.2 Nonlinear Systems

A nonlinear representation of a problem is desirable because actual physical systems are normally nonlinear. A linear model of a system cannot properly represent the complex dynamics of a nonlinear system. A nonlinear system may exhibit very different behavior as its parameters are changed. Because formal mathematical tools for nonlinear system analysis are not as well developed as they are for linear systems, a primary tool of nonlinear analysis is computer simulation. Through computer simulation, the motion of the system can be explored through variations in parameters and external conditions.

2.2.1 Differences between Linear and Nonlinear Systems

The dynamics of simple nonlinear systems can be highly complex, and can be a result of the presence of nonlinearity in the equation instead of the complex interaction of many differential equations. To illustrate this and to demonstrate the differences between linear and nonlinear systems, consider the logistic map (Percival and Richards, 1982; Gleick, 1987). The difference form of the equation, a simple population model, is

$$x_{t+1} = r x_t (1 - x_t) , \qquad \textbf{(2.1)}$$

where x is the population and r is a control parameter that represents the environment. This equation represents the seasonal changes in the population of a species with an annual breeding season. It determines the next season's population, x_{t+1}, using the current population, x_t, and the environment effects, r.

The equation illustrates several interesting phenomena that occur in nonlinear systems. This simple nonlinear equation exhibits sudden changes in behavior as the control parameter is changed. As Figure 2.1 shows, changes in the parameter r change the response of the logistic from steady state through bifurcations to chaos.

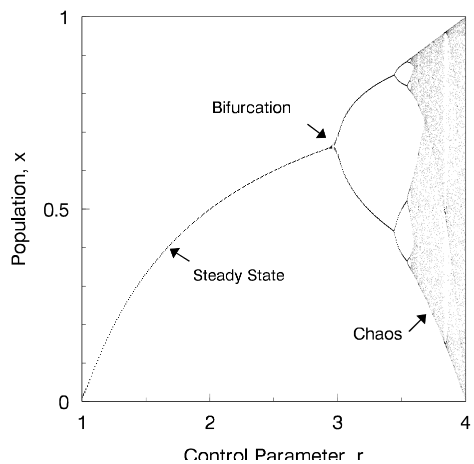

Figure 2.1 Effect of Parameter variation on Logistic map.

In Figure 2.1, the region between 1 and 2.8 is the steady state region. The population, x, can start at any value between 0 and 1. Over time, the population will move to a constant value, which is determined by the environment. For example, if the environmental control factor, r, is equal to 2, then the steady state population will be 0.5. This is the behavior that a single first order linear differential equation would predict.

Above r = 2.9, the population's behavior changes. Between 3.0 and 3.4, the population jumps back and forth between two population states. One year the population will be one value the next year it changes to a different value. With any starting population, the population will settle to an oscillation between the values determined by the control parameter. It never settles to a single value. The transition from a single valued to a double valued behavior is known as a bifurcation.

A single first order linear equation cannot represent this alternating behavior. To represent the two population states, a system of two first order differential equations is required. The complexity of a linear system arises from the interaction of multiple equations. In contrast, the logistic map equation generates the behavior because it is nonlinear.

Returning to Figure 2.1, as r increases to 3.5, the population takes on four alternating states. The bifurcating accelerates as r continues to increase. Finally, the population takes on random values. This phenomenon is known as chaos. This random behavior, the result of a deterministic equation, is a behavior that is not found in linear systems (Farmer and Sodorowich, 1988).

Chaos, however, is not simply a mathematical curiosity. Deterministic chaos has been found in many physical situations including lasers, chemical reactions, plasma physics and turbulence in fluids (Lapedes and Farber, 1987). In fact, chaotic dynamics are quite common and in many cases apparently random behavior is actually chaos (Farmer and Sodorowich, 1988).

2.2.2 Attractors

If undisturbed trajectories of a dynamic system approach a common subset of state space, this subset is called an attractor. An attractor can be a single point, a group of points, a single loop trajectory or a complex locus of trajectories. When an attractor is a single point, it is known as a fixed point attractor. A single loop trajectory is a limit cycle. If the attractor supports chaotic motion, it is a strange attractor.

Attractors are often studied by plotting the trajectories of a dynamic system in phase space. The phase space diagrams bring the attractor structure into view. In phase space, the motion about a fixed point attractor commonly looks like a spiral. Initial conditions that are sufficiently close to the fixed point spiral down to a stop at the fixed point.

Strange attractors are more difficult to visualize. They weave intricate shapes in phase space, never returning to the same trajectory, yet remaining within a subset of the space. To illustrate the motion, Figure 2.2 presents the strange attractor of Lorenz (Gleick, 1987).

It is important to note that strange attractors are produced only in dissipative systems. Energy flux, energy entering and energy leaving the system, must exist. For the attractor to be sustained over time, a balance of energy entering and leaving the system must be maintained.

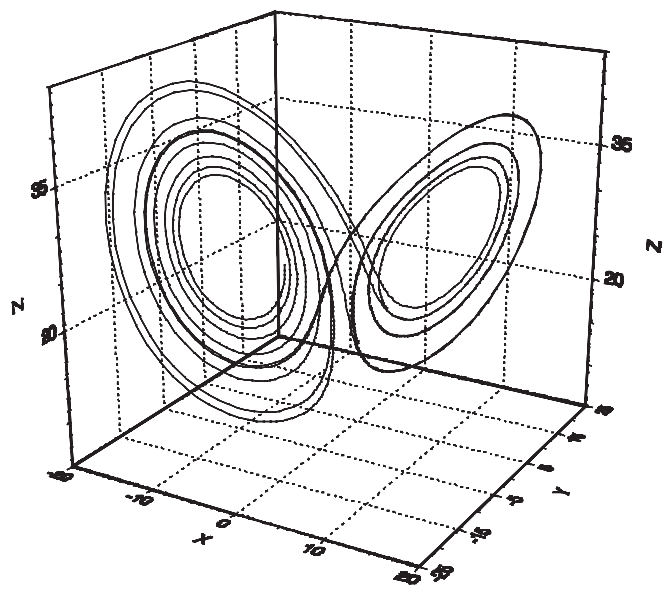

Figure 2.2 Lorenz Strange Attractor

The Lorenz attractor is generated by the following system of equations:

$$\begin{aligned} \frac{dx}{dt} &= 10(y - x), \\ \frac{dy}{dt} &= -xz + 28z - y, \\ \frac{dz}{dt} &= xy - \frac{8}{3}z. \end{aligned} \qquad \textbf{(2.2)}$$

2.3 Stability of Systems

A system whose states will remain bounded for all time is known as stable. To be stated more precisely, any system can be expressed as,

$$\frac{dx}{dt} = f(x,t), \qquad \textbf{(2.3)}$$

where **x** is the state vector and t is time. It is stable if there exists a number $\mathscr{E} > 0$ such that if $\| x(t_0) \| < \mathscr{E}$ then $\| x(t) \| < \mathscr{E}$ for all $t > t_0$ (Brogan, 1985). This is known as Lyapunov stability.

For linear systems, there are many methods for determining the stability of the system. Techniques like the Routh-Hurwitz, Nyquist, Root Locus plot, and Bode diagram are all popular methods (Kuo, 1982).

Nonlinear systems, however, are much more difficult to prove stable. One technique is Lyapunov analysis (Brogan, 1985). This technique centers on the formulation of an "energy" function for the dynamic system. Stability is then proven by showing how this function is bounded or decreases over a particular domain. Currently, finding a Lyapunov function for a general nonlinear system can be very difficult. There are techniques for certain classes of systems, but no general methods for all systems.

In actual systems, the proof of stability is even more intractable than in mathematical nonlinear systems. Many real systems contain nonlinearities that are not mathematically modelled. Also the parameters that are contained in the mathematical model are subject to measurement errors. Because of these problems, proof of the mathematical model's stability does not guarantee the stability of the actual physical system. Stability is, therefore, a matter of system design and experience rather than analytical proof.

The stability of a system is very important. Instability in a physical system at best means the system will have erratic behavior, at worst the system will fail catastrophically. To emphasize this, the controls researcher, Benjamin Kuo, (1982) classifies systems as useful if stable or useless if unstable. This is especially true of automatic controllers. A controller can provide stability to unstable systems or de-stabilize a stable system. Therefore, system stability is an important part of any controller design.

2.4 Non-stationary Systems

A non-stationary system is a system where structural parameters vary over time. For example, in the logistic equation 2.1, if the parameter **r** varied over time, then the system would be non-stationary. Major changes in systems such as the decreasing mass of a rocket in flight are state variables in the system. Other changes, such as the changes in engine performance as a function of temperature, may be included in the model of the system as a state, or they may be simply considered non-stationary effects on the system.

Real systems always have non-stationary parameters. An aging automobile engine is a prime example where many changes in operation occur over time. An easy start on a cold morning with a new car can become a hard start with an older vehicle. Other systems, however, may be much less affected. For example, the pendulum and clock works of a grandfather clock are relatively stable for decades.

The important point in the context of control systems is that the system will change with time and the controls must be capable of safely operating the system through all normal variations. The non-stationary effects are usually compensated by feedback to the controller. If, however, the changes are large or are not measured by the controller, then the controller needs re-tuning.

2.5 Recent Results in Dynamic Systems Theory

Over the last 10 to 15 years there has been a major increase in the study of nonlinear systems. This work has produced some results that are significant to automatic control. This section describes the discoveries that are important to adaptive control. Specifically, the presented findings show that the dissertation objectives are theoretically possible.

The central idea behind all advanced controllers is to improve control by incorporating more physical system knowledge. When the controller incorporates knowledge, it can predict or anticipate the system. In this context, an adaptive controller is simply a controller that gathers system knowledge while it operates. Therefore, for an adaptive controller to be feasible, it must be capable of predicting the system either directly or indirectly, be of reasonable size, and be able to obtain the information from the system. With linear systems, these requirements have been demonstrated. For nonlinear systems, however, it is not immediately obvious that this is possible. Nonlinear phenomena like chaos and bifurcations cast doubt on the feasibility of predicting the system behavior. The remainder of this section presents the evidence that adaptive control is indeed possible.

The first finding is that high dimensional systems can collapse into low dimensional attractors. The Mackey-Glass equation, which due to a time delay in the equation, has a phase space of infinite dimension (Lapedes and Farber, 1987). The system, however, collapses to a much reduced strange attractor. Other systems such as fluid flows have practically infinite state spaces, yet their motion can be represented by low dimensional attractors (Farber and Sidorowich, 1988). Thus, infinite dimensional systems, which are normally analyzed with nonlinear partial differential equations collapse into low dimensional attractors. This is important for two reasons: 1) it is a different view of the nature of nonlinear complexity and 2) nonlinear controllers stand a chance of representing the phenomena.

The first issue is that large systems can organize themselves. The classical view of systems is that complex behavior is due to a high level of complication. In other words, the complex, apparently random behavior of a system results from many degrees of freedom interacting. The chaos result, however, indicates that even systems with few degrees of freedom can exhibit very complex behavior. Even more important is the result that high dimensional physical systems can self organize so that overall system behavior can be described by a low order system. Thus, predictions about systems of high dimension do not require the computation and storage of every degree of freedom of the system. This is illustrated by Lapedes and Farber (1987), where a reduced dimensional nonlinear prediction method can predict the dynamics of an infinite dimensional system. Since a reduced order nonlinear mathematical system can predict an infinite physical system, a nonlinear controller stands a chance of predicting the system and improving the control of the system.

The second important finding concerns the determination or reconstruction of an attractor. Takens (1981) proved a least upper bound of the input dimension necessary to reconstruct an attractor for which a smooth mapping may be formed. If the dimension of an attractor is $\mathbf{d_A}$, then an embedding dimension is needed of less than or equal to $\mathbf{2d_A + 1}$. Lapedes and Farber (1987) indicate that the minimum dimension is the same as the attractor dimension. They also demonstrate that a neural network can approximate the attractor sufficiently for prediction. In fact, the neural network prediction is orders of magnitude better in accuracy than other conventional methods. This means that an attractor may be reconstructed with a reasonably sized controller.

The final finding of interest is probably the most important. The issue is observability. A system is observable if the internal states of the system can be determined from the outputs of the system. Observability is the coupling of the internal states to the output of the system.

Packard *et al*. (1980) demonstrated that an attractor can be reconstructed from a single coordinate by using time delays. The details of the attractor may change, but the topological characteristics or geometrical form remain intact. Any single output of any dissipative dynamical system is sufficient to reconstruct the attractor. Thus, even if all the states of the system cannot be measured, the attractor can be reconstructed by augmenting the input data with delays of the measurable states. This means that all systems are observable. This is important because the proper states needed to form the attractor may be unmeasurable or unknown. Nevertheless, this finding proves that the attractor can still be reconstructed.

Together, the findings prove that adaptive control of nonlinear systems is possible. In spite of the complexities of nonlinear systems, improved control is possible by reconstructing the physical system attractor and using the attractor to predict the system.

2.6 Control of Physical Systems

Control of dynamic systems is the regulation of a system to perform a useful task. The term *automatic control* is also commonly used to emphasize that the control is done by equipment rather than manually.

Control systems are divided into two broad classes; open-loop and closed-loop. The input to both systems is the set of desired goals, which are determined external to the automatic controller. Figure 2.3 and Figure 2.4 illustrate the basic structures of open-loop and closed-loop control systems.

Figure 2.3 Open-loop control system

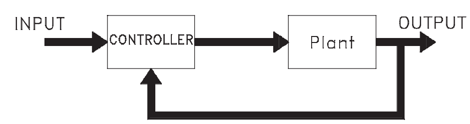

Figure 2.4 Closed-loop control system

Open-loop control requires that the controller contain sufficient knowledge of the system and its inputs to control the system without measuring the system or plant output. The closed-loop approach uses information from the output of the plant to provide control with less *a priori* knowledge. Closed-loop or feedback control systems are generally better able to accommodate unexpected disturbances to the system than open-loop controllers.

2.7 Classical Control

Classical control techniques were developed in the 1930s. Most of the techniques concentrate on linear constant coefficient systems with one input and one output (Brogan, 1985). Common tools are Laplace and Z-transforms and transfer functions.

Surprisingly, classical controllers comprise the bulk of the controllers in use today. The two most common type of controllers are the bang-bang and PID (Proportional-Integral-Derivative). They illustrate the level of sophistication of most present controllers.

The best example of a bang-bang controller is the common home thermostat. In heating mode, if the temperature is below a certain temperature, the furnace is activated. Once the temperature is sufficient, the thermostat shuts the furnace off. The main design constraint is the need for some hysteresis or dead band around the set point to prevent rapid on/off cycling. The primary advantage is low cost and simple operation. The primary disadvantage is that the system is never exactly at the set point due to the need for a dead band. Common applications include anti-lock brakes, sump-pumps, ovens, refrigerators and many other systems.

The second type is a PID (Proportional-Integral-Derivative) controller. This controller is generally used where a smooth range of control is available and more precision is needed than a bang-bang controller can provide. The control action of a PID, shown in Figure 2.5, is the weighted sum of the error, the derivative of the error and the time integral of the error.

These controllers are used heavily in the process and manufacturing industries. PID controllers are also used in conjunction with many other controllers. The main advantages are relatively inexpensive components and the wide familiarity of users with their operation. The main disadvantages are that selection of the weighting coefficients can be difficult, and changes in the plant conditions can place the controller in conditions where the coefficients are inappropriate.

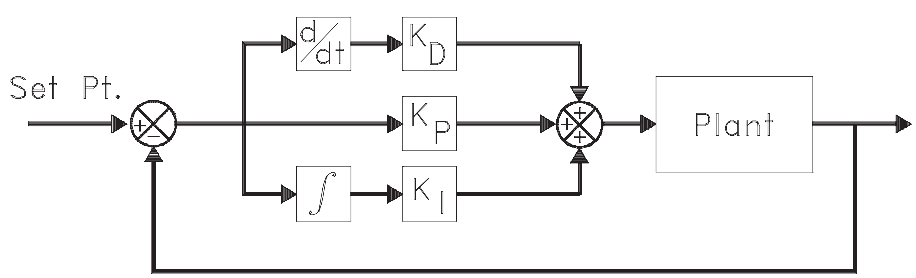

Figure 2.5: Proportional-Integral-Derivative (PID) Controller

2.8 Modern Control

Modern Control refers to the work in controls since the late 1950s and early 1960s (Brogan, 1985). Central to many of these techniques is a time-domain analysis using state variable models. This section focuses on two of these techniques; optimal and adaptive control. Also included is discussion of expert/rule-based control.

Optimal control (Brogan, 1985) refers to a variety of methods. All the techniques work to bring a measure of performance to an optimum value. Therefore, central to all the methods is an optimization technique. Common optimizers include dynamic programming, linear programming, and variational calculus techniques. Generally, due to its computational complexity, optimal control is performed off-line. Planning the path of a spaceship or scheduling flights for an airline are examples of off-line optimal control. The requirements of optimal control include a model of the system and a scalar measure of performance.

An adaptive controller combines on-line parameter estimation with on-line control (Goodwin *et al.*, 1984). The idea is to design a controller that can self-adapt to changes in the system. Examples of adaptive controllers include Gain Scheduling, Model Reference Adaptive Systems (MRAS), Self-Tuning Regulators (STR), and other learning controllers (Astrom, 1983; Wittenmark and Astrom, 1984; Cramer, 1990; Narendra and Thathachar, 1974).

Expert control or rule-based control is the application of expert systems or production systems to control. The hallmark of these techniques is that the control of the system is described in linguistic terms instead of mathematically. Expert control systems are generally used as supervisors, regulating controllers rather than directly controlling the physical system. Expert controllers are also popular for governing the adaption process of adaptive controllers. Applications and design details for expert systems are given in Astrom (1986), Bernard (1988b), and Handelman *el al.*, (1989).

2.9 Summary

This chapter described nonlinear systems and control of physical systems. Topics discussed include differences between linear and nonlinear systems, bifurcations, attractors, and chaos. Dynamic system stability was presented, including the definition of Lyapunov stability and the importance of stability to dynamic systems. Recent results in nonlinear systems theory were used to show that adaptive control of nonlinear systems is theoretically possible. The fundamentals of physical systems control and common methods of physical system control were also presented.

CHAPTER III

ARTIFICIAL NEURAL NETWORKS

3.1 Introduction

Artificial neural networks are methods of computing that are designed to exploit the organizational principles that are felt to exist in biological neural systems. By its very nature, the study of neural networks is interdisciplinary with investigators from engineering, physics, and computer science joining scientists from neurobiology and psychology.

Although theoretical explanations of the brain and thinking can be traced to early philosophers like Plato (427-347 B.C.) and Aristotle (384-322 B.C.), much of what today is known as artificial neural networks is the result of the last 30 years of study. The fundamentals of modern neural network computing can be traced to the early 1940s research of W.S. McCulloch and W.A. Pitts (Anderson and Rosenfeld, 1988). Through the 1950s[1] and 1960s there was considerable growth in the field until the Minsky and Papert (1969)

[1] The design of the modern digital computer was also greatly influenced by early neural network research. See "The Computer and the Brain", Von Neumann, 1958.

book, Perceptrons, demonstrated the limitations of the artificial neural networks of the day. In the early 1980s, there was a renaissance, and with the renewed interest came solutions to problems elucidated by Minsky and Papert (1969).

The motivation for study of neural networks is twofold. For biological scientists and psychologists, artificial neural networks provide a means for interpreting and analyzing the results of studies of biological neural networks. For the engineer or physical scientist, the motivation is to bring biological neural network capabilities to technological systems. Abilities like image analysis, character recognition, reasoning with incomplete and fuzzy information, and generalization of examples are all desired capabilities.

This chapter discusses the structure and operation of neural networks. Neural networks in general are presented, but the primary focus is layered feedforward networks. Section 3.2 discusses the fundamental properties and functions of neural networks. Section 3.3 demonstrates the importance of nonlinearity to the operation of a neural network and discusses the most common nonlinear functions. The ways in which data are presented to a neural network are the topic of Section 3.4. Training a neural network is described in Section 3.5. The last two sections present techniques for training a neural network: back-propagation in Section 3.6 and reinforcement learning in Section 3.7.

3.2 Properties of Artificial Neural Networks

The primary inspiration from biological systems is the composition of the computing elements and the structure of the interconnections[2]. Presently, the fundamental unit of computation is felt to be the neuron. Biological neurons vary greatly, but all neurons have common elements.

The common elements are a dendritic tree for input, a cell body, and for output an axon and axonic tree. The dendritic tree is the input side of a neuron and collects signals from the axon trees of other neurons. The cell body maintains the operation of the entire cell and provides a processing action on input signals. The resulting signal is then passed down the axon to other neurons. The connections between neurons are known as synapses and vary in strength.

Artificial neural network models vary greatly depending on the application. Biological studies emphasize operational characteristics of neurons and result in some of the most complex models. Engineers and

[2]The operation of biological neural networks is not completely understood. The neuron anatomy is correct, but the formation of synapses and the function of neurons in a network is not fully understood.

computer scientists, however, are more interested in the computational capabilities. This dissertation is concerned with the latter emphasis in neural networks. From the computational view point, a neuron is commonly modeled as illustrated in Figure 3.1. When neural networks are discussed from the computational view, a neuron is usually called a node to distinguish from biological neurons.

This view of a neuron consists of two main parts: a collection function

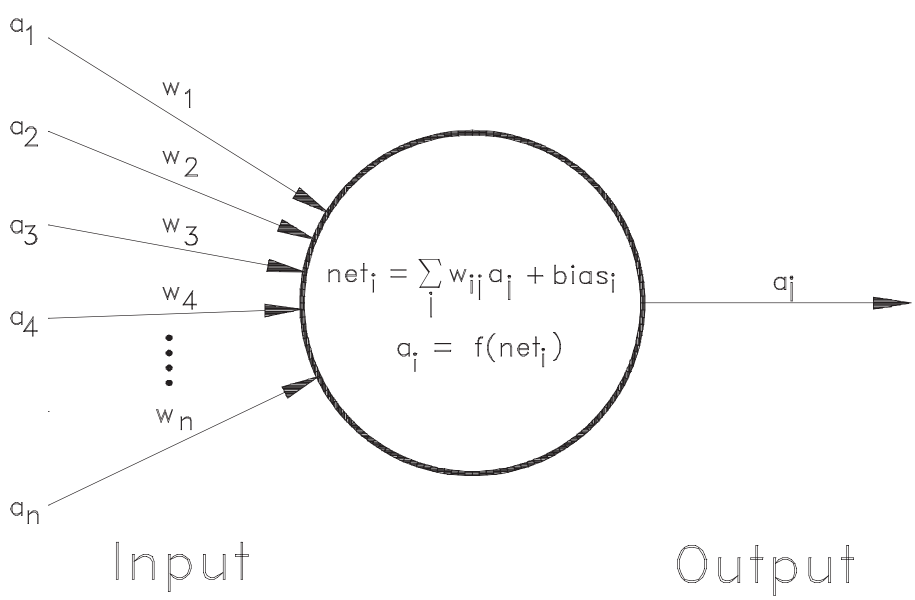

Figure 3.1: Computational model of neuron.

and an activation function. The collection function is inspired by the dendritic tree and the associated synapses. Its purpose is to gather the output from neighboring nodes and to determine the contribution of each input to the state, net_i, of the node. In Figure 3.1, the collection function is the equation:

$$net_j = \sum_i w_{ij} a_i + bias_j, \tag{3.1}$$

where a_j is the output of other nodes, w_{ij} are the connection strengths from input node i to the current node j, and the bias term is the node's offset. The activation function, representative of the information processing component of a neuron, uses the node state, net_i, to calculate the output of the node. This output serves as input to other nodes. The activation function, $f(net_i)$, is the topic of Section 3.3.

Neural networks store information in the structure of their interconnections. For the above model, this means that the weights, w_{ij} and the $bias_i$ are adjusted to store information. The weights are connections between nodes. By adjusting the weights, the influence of each input node on the node's processing is regulated. The bias is often considered a connection to an input node that is always fully activated. Its purpose is to set a threshold or offset value for the node processing.

Artificial neuron building blocks are combined to produce neural networks. The nodes of a network are normally broken into three groups:

input nodes, hidden nodes, and output nodes. The input nodes receive input from outside the network. Output nodes are the outputs from the network. Occasionally a node may be both an input and output node.

The other nodes, which receive and supply input only to other nodes in the network are known as hidden nodes. These nodes are not directly visible from the network input or output, but are extremely important to the operation of the network. They have the ability to form combinations of the input, which can be the foundation of later processing in the network.

3.2.1 Information Storage

As a natural result of storing information in weights, the information or knowledge is distributed. This is one of the most important differences between neural network computations and a standard Von Neumann style digital computer. A standard digital computer stores information as words or tokens in the memory. Each token is a concentrated highly specific piece of data. The loss or corruption of even one token can result in a complete failure of the computer. Instead of focusing data into a token, a neural network stores the knowledge in a distributed form. The knowledge is stored across many weights with each weight contributing to many parts of the knowledge. Thus, the information is stored in the cumulative interaction of many nodes and their associated weights. In contrast to a Von Neumann digital computer, the loss or corruption of a weight simply degrades the performance of the network and does not result in failure of the network.

Another important difference is that information in a neural network is stored in an associative memory instead of an addressed memory. In an associative memory, data is retrieved by its similarity to the input data. Common digital computers use addressed memory. Each memory location is given a label, and the contents of the location are accessed by providing the datum label or address to the memory. In contrast, when a neural network is presented a possibly incomplete or damaged version of the desired data, the network locates the most similar data in memory. As such, some networks are designed simply to be associative memories. In other network architectures this ability is used to generalize between training examples.

3.2.2 Universal Approximation with Back-Propagation Networks

Manual selection of the weights is usually possible for only very small networks. Therefore, a learning algorithm or method for adjusting the weights is an important part of every neural network. The learning algorithm trains the network by learning from examples. One of the most popular neural network training techniques is back-propagation. Usually back-propagation refers to a neural network architecture, which is a layered network without feedback. This architecture is illustrated in Figure 3.2.

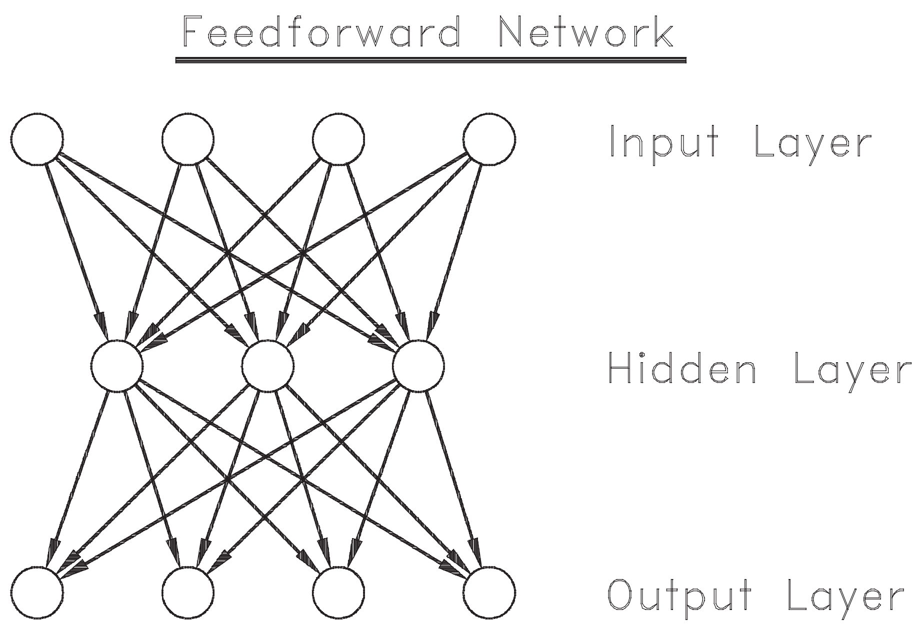

Figure 3.2: Layered Neural Network.

The size of a back-propagation network varies. There can be more hidden layers and/or more nodes in any layer. The general structure, however, remains the same. Connections are only between layers, such that information flows from the input side, layer by layer, to the output. Normally, there are no connections between nodes in a given layer or from a lower layer to a previous layer. Occasionally, there are intra-layer and/or inter-layer feedback connections. Networks of this type are known as recurrent networks.

A common use of a back-propagation neural network is as a nonlinear functional map. The network is considered a black-box, which maps the input space into the output space (Lapedes and Farber, 1987; Farmer and Sodorowich, 1988). This use is also one of the ones that is best supported by theory. Stinchcombe and White, (1989) have published a series of papers proving that if the activation function is a particular type, then a back-propagation network can approximate any function. A neural network with this capability is known as a universal approximator. The universal approximation theory was further expanded by Kreinovich (1991), who proved that any arbitrary nonlinearity is sufficient to approximate any function with a feedforward neural network. For further discussion of activation functions see Section 3.3.

In closing this section, a short description of how a neural network approximates any function is in order. First, it should be noted that there are other universal approximation methods. Common examples are: 1) Taylor series, which use polynomial approximation and 2) Fourier series, which use the sine and cosine trigonometric functions to approximate any continuous function. Thus, it really should be no surprise that any nonlinearity is sufficient to form the foundation of a universal approximator.

In a neural network with a sigmoid activation function (See section 3.3), the network is performing an operation similar to a multi-dimensional spline approximation (Farmer and Sodorowich, 1988; Lapedes and Farber, 1987). The network forms a surface that is the best fit to the example cases. Thus, the generalization/interpolation capability is a natural result of surface estimation. With this understanding, a caution for the use of neural networks is highlighted: Sufficient data is required to shape and orient the surface in space properly. If only a few training samples are used, the network will learn them, but the approximations between them may not be as expected.

3.3 Activation Function Selection

The activation function is the component of an artificial neuron that determines the node's output based on the current inputs to the node. In early neural network models, the activation function was a simple threshold. If the sum of the node's inputs was greater than the threshold, then the output was one. Otherwise, the output of the node was zero.

The most important facet of the activation function is that it is not linear. Nonlinearity is the key to the hidden nodes forming an independent representation. To show the importance of the nonlinearity, consider a four layer neural network with the dot product collection function from Figure 3.1. With this collection function, the weights for a layer can be grouped together as a matrix. Each row in the matrix contains the weights for a node. The number of rows in the matrix equals the number of nodes in the layer. The number of columns is equal to the number of nodes in the previous layer. With this representation, a four layer neural network is represented with three weight matrices, $\mathbf{W_1}$, $\mathbf{W_2}$, and $\mathbf{W_3}$ where $\mathbf{W_1}$ maps the input to the first hidden layer, $\mathbf{W_2}$ maps the first hidden layer to second hidden layer, and $\mathbf{W_3}$ maps the second hidden layer to the output layer. A four layer network with an activation function, **f**, can be written in matrix notation as:

$$N = f(\, W_3 f(\, W_2 f(\, W_1 I))) \,, \tag{3.2}$$

where **N** is a column vector that is the network output and **I** is a column input vector. First, recall that any linear mapping can be represented by a matrix. Therefore, if the activation function, **f**, is a linear function, it can be represented as a matrix, **Q**. Substituting for **f**, in equation 3.2 yields:

$$N = QW_1 \, QW_2 \, QW_3 I \, . \tag{3.3}$$

Noting that any product of linear matrices may be combined to form a single matrix allows equation 3.3 to be reduced to a single matrix, $\mathbf{W_{total}}$. Equation 3.3 becomes:

$$N = W_{total} I \, . \tag{3.4}$$

Thus, the entire network can be reduced to a single linear transformation and the network is only able to solve linearly separable problems.

If the activation function is nonlinear, then the reduction to a single matrix is not possible and the network is able to form nonlinear representations. Thus, the nonlinear activation function is the key component to a neural network's universal approximation capability. The most common nonlinearity is the sigmoid function:

$$f(net_j) = \frac{1}{1 + e^{-net_j}} \, , \tag{3.5}$$

which normally is used with the collection function, which was presented in equation 3.1. The hyperbolic tangent is equivalent to equation 3.5 and is also commonly used.

Other functions may be used. Lapedes and Farber (1987) describe a network using sine as the activation function. The result is a generalized Fourier series network with the improvement over a traditional Fourier series that the component frequencies are variable instead of set on fixed intervals.

Besides the above global representations, local representations are also used in neural networks. One local representation is a radial basis function, which is a gaussian bump (Waltz and Fu, 1965; Barto, 1989; Farmer and Sodorowich, 1988; Hartman *et al*., 1989; Saha and Keeler, 1990; Specht, 1990). A local representation is desirable since the modification of one portion of space does not affect other areas. This locality is also the Achilles heel of radial basis functions, since the number of gaussian bumps increases geometrically as the dimensionality of the space increases. In response to this problem, Hartman and Keeler (1991a) have studied a semi-local activation function, which is essentially a gaussian line.

At present, the selection of an activation function is more art than science and is the subject of much research. The important point to remember is that any nonlinear function will provide the network with the ability of universal approximation (Kreinovich, 1991).

3.4 Data Representation

The representation of the input and output data is the topic of this section. Although a back-propagation neural network forms its own internal representation in the hidden units and is capable of mapping any input space to any output space, the representation of the input and output data can ease the task for the network. Topics discussed in this section include preprocessing of input data, presentation of data to the network, and the representation of time for both feedforward and recurrent networks.

Although a back-propagation neural network can form any mapping, preprocessing of the input data can simplify the network's internal mapping. For example, if the desired network output requires the square of an input line, then augmenting the input by the square of the needed input line will remove the need for the neural network to perform the squaring operation. This augmenting can be stretched to the extreme where all the function components required for generation of the output are precomputed and the task of the network is to learn the proper combination (Lapedes and Farber, 1987; Miyamoto *et al.*, 1988; Nilsson, 1965). A neural network of this type is known as a functional network.

Even if no precomputation or combination of data is performed, there is still the issue of how the data will be presented to the network. For binary tasks, the common technique is to have one node for each bit of input data and simply set the input node to full activation or minimum activation.

Continuous value input can be accomplished in a variety of methods. The most obvious method is to scale the input to the activation range of the input nodes, and then set the input node's activation to the value indicated by the input scaling. Another method is to use a group of input lines for each value and set up a temperature scale arrangement. The temperature scale method simply activates more input nodes as the input value increases.

An extension of the temperature scale method is to use window functions. See Figure 3.3. A given input value will stimulate more than one input node, but each node is activated to a different degree. This method is most similar to what is seen in biology (Mead, 1990). For example, photoreceptive cells in the eye are more sensitive to certain wavelengths of light and less sensitive to other wavelengths.

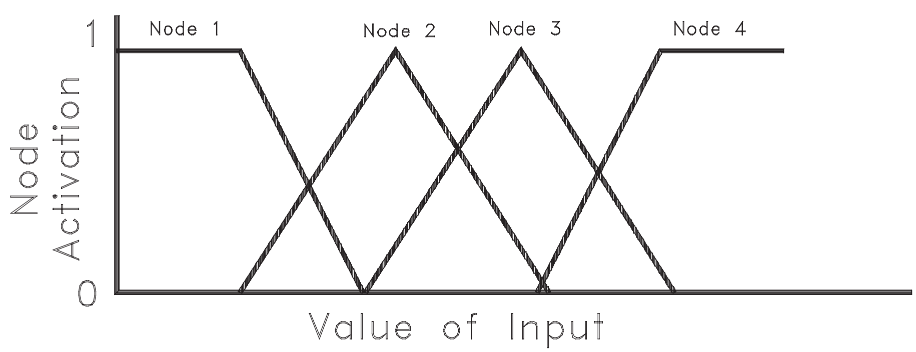

Figure 3.3: Input Window Functions

The representation of time is important to applications like speech recognition, signal processing, and automatic controls. Presently, the two methods for including time in the network input are spatial expansion and recurrent networks. Interestingly, the human body uses both methods: spatial expansion in the cochlea (Mead, 1990), and recurrent networks for muscle control.

Spatial expansion uses time delays to spread the time history of the signal over many nodes. Figure 3.4 illustrates a fixed delay for step, but the delays need not all be the same. For example, Mead (1990) has developed hardware that uses geometrically increasing delays to emulate the human cochlea.

The major disadvantage to the spatial representation is that as the

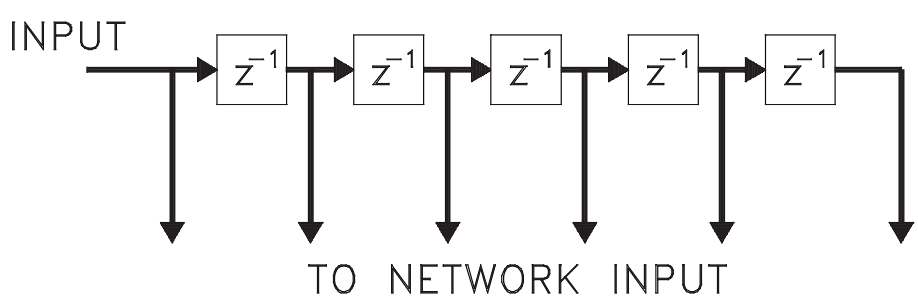

Figure 3.4: Spatial Expansion of Time History of a Signal

size of the network is increased, it may be difficult to determine the number and size of delays required. The number of inputs to a neural network cannot be increased indiscriminately due to a computational complexity limitation. A neural network requires $O(N^2)$ operations, where N is the number of nodes in a layer. Thus, if the network input layer is doubled in size, then the computation for that layer increases by a factor of 4. The computations, however, are inherently highly parallel and with the advent of highly parallel computer hardware this problem is reduced. In any event, it is still generally better to seek the minimum computation load.

The number of time delays in the input depends on the time scales present in the time history. Sufficient time delays are required to span the longest cycles in the systems. Simultaneously, the length of delays must be short enough not to miss the short duration cycles in the data stream. Thus, this problem is the most severe when wide variations in time scale are present in the data.

A recurrent network solves the representation of time in a completely different way. Time is represented implicitly by its effect on processing instead of explicitly in the input data stream (Elman, 1988). In a recurrent network, the past activation states of the network affect the network in the present. The input data act as context information which guides the processing. This has the benefit that the duration of the time history is not limited by the number of delay units. To accommodate the shortest cycles in the input data the network simply needs a faster cycle time than the shortest cycle in the data. Recurrent network dynamics, however, are significantly greater in complexity than feedforward networks and presently there is less experience with their operation.

3.5 Learning Algorithms

A learning or training algorithm is a core component of most neural networks. Although a learning algorithm is needed to determine the connection weights, a primary benefit of the learning algorithm is the mitigation of the knowledge acquisition bottleneck.

The knowledge acquisition bottleneck is a term coined by the artificial intelligence community to describe the difficulty of determining the relevant knowledge and incorporating it into a rule base or expert system. This problem is not limited to the artificial intelligence community, but it also is a common problem for scientists and engineers. For example, consider the time and effort required to conduct a test or experiment and process the data to yield a mathematical rule or equation.

A neural network solves the problem in the same way that humans and animals often solve the problem, by learning from examples. Within this context of learning by example, there are many methods for training a neural network. Neural network training procedures are divided into three categories: supervised learning, unsupervised learning, and reinforcement learning.

Supervised learning is learning under the direction of a teacher. The network is supplied with both input and output examples and the learning algorithm adjusts the connection weights to match the mapping. Character recognition is a good example of this type of learning where training examples consist of input examples and the appropriate network output. A back-propagation network is a prime example of a supervised learning network (Rumelhart and McCelland, 1986).

The second learning procedure, unsupervised learning, requires the network to determine its own proper classification. Training samples consist of input examples only. The learning algorithm self-organizes or clusters the data into classes. The main motivation for unsupervised learning is that biological neural networks appear to self-organize. Kohonen (1988b) feature maps and Adaptive Resonance Theory networks (Grossberg, 1988) are examples of unsupervised learning networks.

Reinforcement learning lies mid-way between supervised and unsupervised learning. Input examples are supplied to the network and a scalar signal of how good the network is performing is used to adjust the network. Normally, a series of examples is presented, where each input produces a network response. At the end of the input series, the scalar input of performance is supplied and the network adjusts itself to improve its performance.

The difference between supervised learning and reinforcement learning is that supervised learning has an external teacher to tell the network how to adjust, while reinforcement learning must decide the proper adjustment from a good/bad signal. No information of how to improve is provided.

The next two sections present in more detail the supervised learning technique, back-propagation, and reinforcement learning. The back-propagation section, Section 3.6, presents the gradient descent learning algorithm. Section 3.7 further describes reinforcement learning. Topics include the Michie and Chambers (1968) boxes algorithm, adaptive critics, and temporal difference methods.

3.6 Back-Propagation Learning

The term *back-propagation neural network* is a generic title for a method of training. It combines a means for distributing error among nodes with a gradient descent learning algorithm. The back-propagation label describes the way in which the neural network distributes the error. The error signals flow in an opposite direction than the activation signals. Figure 3.5 illustrates the normal forward flow of information and the reverse flow of error information that is used to adjust the weights. Back-propagation usually

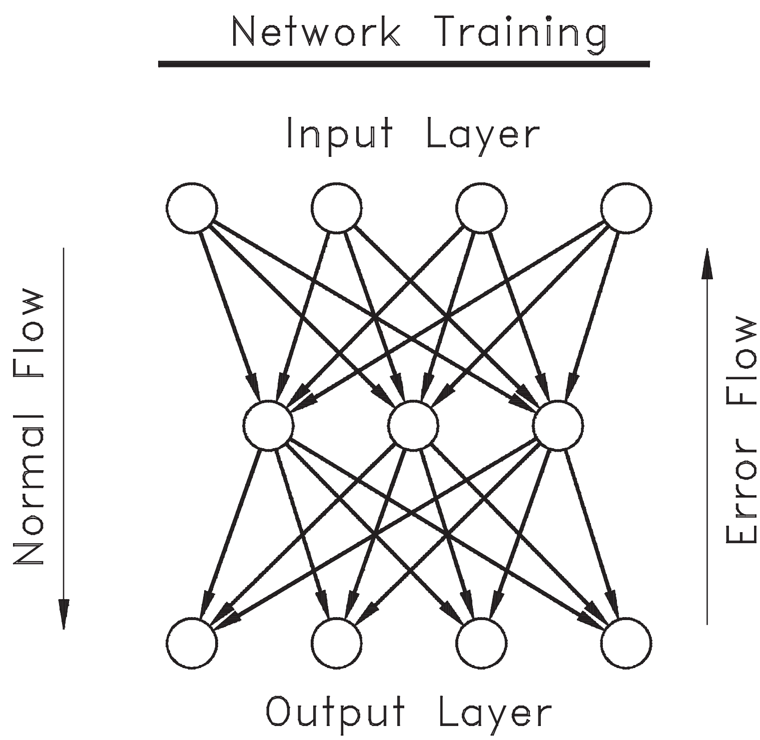

Figure 3.5: Flow of information in feedforward back-propagation neural network.

describes a simple feedforward network, but the method is flexible and has been adapted for training recurrent networks (Elman, 1988; Jordan and

Rumelhart, 1990b; Jordan, 1989; Jordan, 1988; Pineda, 1987a; Pineda, 1987b; Pineda, 1988; Pineda, 1990; Pearlmutter, 1988; Pearlmutter, 1990; Robinson and Fallside, 1987).

3.6.1 Training Algorithm Development

The back-propagation learning algorithm is the application of the gradient descent nonlinear optimization technique (Rumelhart and McCelland, 1986; Rumelhart and McCelland, 1988; Luenberger, 1984)[3]. The usual objective function is the sum squared difference, over all the training patterns, between the network outputs and the desired outputs. In matrix notation the objective function, E, is:

$$E = (T - N)(T - N)^T, \tag{3.6}$$

where **T** is a p x m matrix of desired output patterns and **N** is a p x m matrix of network outputs. The index p is the number of patterns in the training set and m is the number of output nodes in the network.

The output of the network, **N**, is a function of the network weights, $\mathbf{W_1}$, $\mathbf{W_2}$ **. . .** $\mathbf{W_l}$, of all the network layers l. To minimize the objective function,

[3]Common in the literature, the description of the back-propagation is done using scalar variables and summations (Rumelhart 1986, Wasserman 1989). Nevertheless, to facilitate the development of the distal-error training algorithm, this presentation of back-propagation mathematics uses matrix formulation.

E, gradient descent specifies that the weights are to be adjusted according to:

$$\Delta W = -\alpha \frac{\partial E}{\partial W}, \qquad \textbf{(3.7)}$$

where α is a proportionality constant. Expanding equation 3.7 using the chain rule gives:

$$\textbf{(3.8)} \qquad \Delta W = -\alpha \frac{\partial E}{\partial f} \frac{\partial f}{\partial x} \frac{\partial x}{\partial W}.$$

The last term in equation 3.8 is the output of the previous layer. The center term, for single degree of freedom activation functions, is the ordinary derivative of the activation function. The first term, however, varies depending on the layer in the network. Thus, equation 3.8 is normally expressed as:

$$\Delta W_l = \varepsilon \Gamma_l \frac{df}{dx} A_{l-1}, \qquad \textbf{(3.9)}$$

where Γ_l is the layer error matrix and $\mathbf{A_{l-1}}$ is the matrix output from the previous layer. Equation 3.9 is known as the weight update equation where for output units:

$$\Gamma = (T - N). \quad \textbf{(3.10)}$$

The hidden units receive their error by summing their contribution to the errors on all the following layer. This is accomplished by multiplying each node's error in the following layer by the interconnection weight and summing up all these back-propagated errors. In matrix notation, Γ_l is:

$$\Gamma_l = \Gamma_{l-1} W_{l-1}^T. \quad \textbf{(3.11)}$$

3.6.2 Disadvantages of Back-propagation

One disadvantage of gradient descent is the problem of local minima. Because the method uses only neighborhood information to determine its direction, it may move to the bottom of a minimum, which is not the global minimum. This is illustrated in Figure 3.6.

In practice, however, local minima are rarely a problem. The local minima tend to have ravines by which a gradient descent method can escape. The problem also can be mitigated by using a few extra hidden units to give the network some more degrees of freedom to escape or by simply restarting the algorithm at a different starting location. Nevertheless,

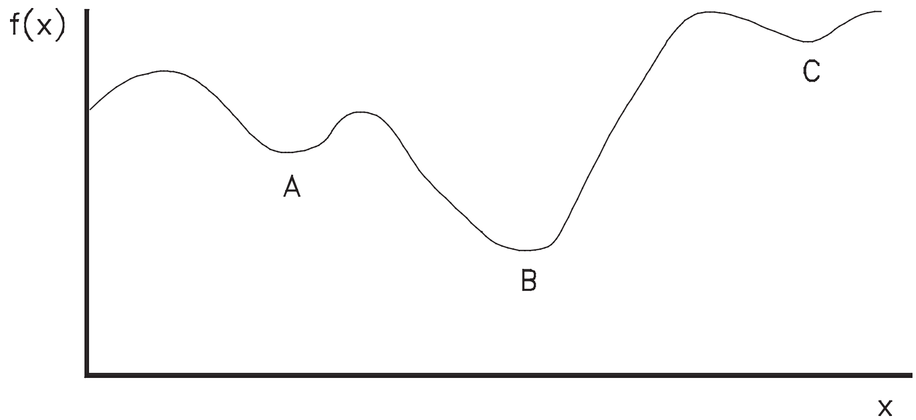

Figure 3.6: Minima on a curve. Positions "A" and "C" are local minima, position "B" is the global minimum.

local minima problems appear in some applications and neural network researchers keep local minima in mind when training a back-propagation network.

Another disadvantage to back-propagation training is slow training. Gradient descent methods are notoriously slow (Luenberger, 1984). Conjugate gradient and other methods have been used with success, but the most common solution is to add a momentum term to the update equation:

$$\Delta W_l(n+1) = \varepsilon \Gamma_l \frac{df}{dx} A_{l-1} + \beta \Delta W_l(n)\,, \tag{3.12}$$

where n indexes the presentation cycle and β is a constant that determines the effect of past weight changes on the current change. The primary effect of the momentum is to calm oscillations in the weight updates. This is common when traversing a steep walled ravine with a gently sloping floor. The steep sides induce oscillations across the ravine, which slow the progress down the ravine. The momentum term also has the side benefit of helping the training algorithm move through small local minima.

In spite of the known problems with back-propagation, the wide versatility of the method, combined with proofs of universal approximation capabilities, result in one of the most popular and widely used neural network techniques.

3.7 Reinforcement Learning

The main difference between a supervised training technique and reinforcement learning technique is the quality of the error signal. A supervised training algorithm requires an externally supplied target value for the output. Therefore, the training algorithm knows the magnitude and direction of the current output from the desired output. A reinforcement learning algorithm, however, has a less informative error signal. It receives only a good/bad signal without a target value. Therefore, the reinforcement training algorithm must determine the proper direction in which to adjust itself.

Another difference between supervised learning and reinforcement learning is the frequency of the error signal. A supervised training algorithm has a target value for each network output. In contrast, a reinforcement learning algorithm usually receives an single error signal after a series of steps. Thus, it must learn the sequence of steps that returns the best reinforcement signal.

A main motivation for reinforcement learning is that it is the primary learning method of biological systems. Animals learn and adapt daily with only reinforcement type error signals. Reinforcement learning studies, therefore, seek to capture similar capabilities in artificial systems. Just as artificial neural networks are patterned after biological neural networks, reinforcement learning systems strive to emulate animal learning. Thus, experiments of Pavlov and Skinner (Holland and Skinner, 1961) and others, which detail the fundamentals of behavior in animals, are important guides.

An early approach to this problem is the boxes solution of Michie and Chambers (1968). The basic idea is to break the environment space into many small sub-spaces. Then using trial and error, decide what action should be taken in each sub-space or box such that the final outcome is optimal. After training, an action is associated with each box such that the maximum reinforcement is returned for any initial condition.

Barto *et al.,* (1983) built on this foundation and added another element to the system. The new element, an adaptive critic, greatly increased the speed of learning. The purpose of the critic is to estimate the present chance of a positive final outcome for each box. This aids the learning because the critic provides a reinforcement signal when one is unavailable from the environment. The critic provides a measure, which predicts the future performance of the system. Thus, by maximizing the critic's measure the final return is maximized.

Werbos (1991) describes the universal importance of the critic's function to fields ranging from behavior psychology to physics. For illustration, a good domain is business theory, where the purpose of the critic would be to determine the present value of all assets. The adaptive critic provides a present estimate of what the future outcome will be. This reduces the problem to a local one (in time) of selecting the best action to yield the best outcome. Thus, by maximizing the present value the final return will be maximized.

Although there are a variety of ways to build a critic, the method of Barto *et al*., (1983) is particularly interesting. They used a method known as temporal difference. The goal of a temporal difference technique is to estimate the expected value of the final outcome given the current state, but it does this by comparing two successive predictions of the expected value (Sutton, 1988). By working to minimize the difference between successive predictions, a temporal difference method enforces temporal smoothness of the estimate. Just as relaxation techniques solve a partial differential equation by spatial smoothing, a temporal difference technique solves the future value problem through temporal smoothing.

The fundamental ideas of reinforcement learning were presented. For more discussion of reinforcement learning and its application to control, see Section 4.6.

This chapter discussed the structure and operation of artificial neural networks. Neural networks in general were presented, but the primary focus was layered feedforward networks trained with back-propagation. Other topics included: the importance of nonlinearity to neural network operation, data presentation, and neural network training with back-propagation and reinforcement learning. The section provides a foundation in neural networks, which is necessary to understand the techniques present in this dissertation.

CHAPTER IV

PREVIOUS NEURAL NETWORK CONTROLLERS

4.1 Introduction

Automatic control is a natural application for artificial neural networks. The necessary infrastructure of sensors and actuators is normally already part of the system. Thus, only the actual controller hardware and/or software needs to be modified.

This chapter will concentrate on back-propagation and its variations for control applications. Other artificial neural network techniques have been applied to control, but the back-propagation techniques are more fully developed. Although the primary focus is back-propagation, significant contributions from other artificial network techniques will be included where they are appropriate.

For automatic control, a back-propagation neural network can be considered an adaptive black box. The neural network has the advantages of a full nonlinear representation and a well developed learning algorithm.

A main issue for neural network control research is the question of how to adapt the network. This is illustrated by Figure 4.1.

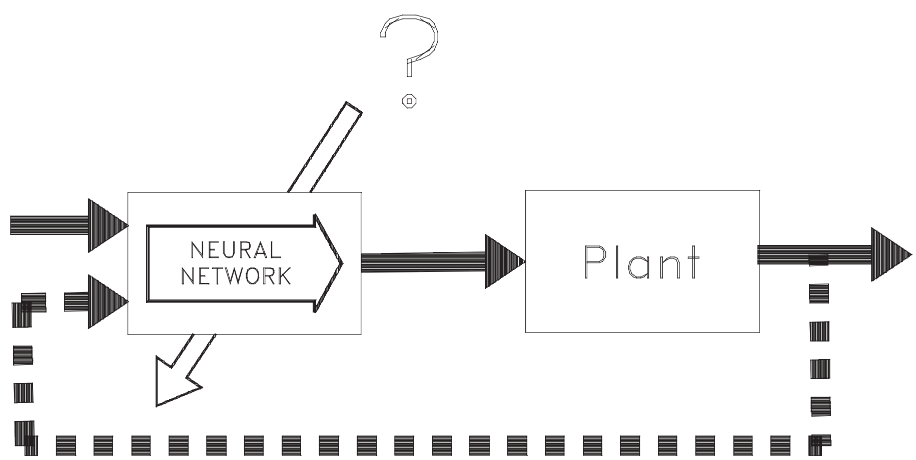

Figure 4.1: The problem of adapting the weights of a back-propagation network in a control application. (Dotted line indicates feedback is optional)

The controller is to drive the plant output to a desired state. The problem is that the plant separates the desired plant outputs from the network's output layer. The difference at the plant output cannot be directly used to train the network because of phase problems. This problem is known as the distal-error problem. Consider a chemical plant, which has piping delays. A control action taken at a particular instant will not affect the process until material has sufficient time to travel though the pipes. If adjustments to the weights are made without considering the delays, an unstable controller will be the most likely result.

The target outputs can be distant in space, time, or both. The above chemical plant is an example of an error that is distant in time and space. The phase problem is a timing problem. There are error signals at each time step; the problem is matching the output errors to the proper control inputs. The spatial component of the problem is determining the influence of each input on the outputs. Some solutions for solving the distal-error problem are presented here.

This chapter presents various neural network methods for control. The objective of this chapter is to provide a foundation in neural network control. This is necessary to understand the proposed architecture and its relationship to other neural network controllers. Sections 4.2, 4.3, and 4.4 present controller designs, which motivate controllers in later sections. The controllers in the these sections also are helpful to introduce neural control to readers unfamiliar with the techniques. Sections 4.5 and 4.6 present important ideas, which are critical to understanding later chapters. The final section, optimal neural network control, is presented to position the research of this dissertation with respect to optimal control methods.

4.2 Copying an Existing Controller

One method for training a neural network is simply to copy an existing controller. This is possible because a back-propagation network is capable of universal approximation. At first, this method seems ridiculous since a controller is already available. The existing controller, however, may be impractical to implement. It could require a person on a missile or it may be too computationally intensive and require a supercomputer to execute in real time. Barto (1989) also mentions that a neural network may be able to provide the control using an input representation that is easier to measure than the representation required by the known controller.

Figure 4.2 illustrates how a neural network may be trained to copy an existing controller. The technique uses the universal approximation capabilities (See Section 3.2.2) of a neural network to approximate the existing controllers mapping from input to output. The neural network is simply placed in parallel to the existing controller and corrected to follow the controller. Once the neural network is trained, it can replace the existing controller. When using this technique, it is important to train the neural network throughout the operating range to ensure robust operation over the entire domain.

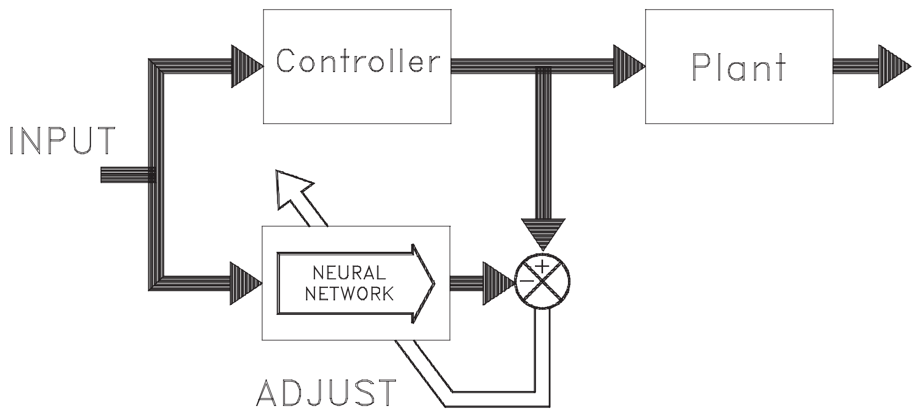

Figure 4.2: Training a neural network controller with an existing controller.

4.3 Delta/Dual Control

A neural network may be combined with another controller to enhance the overall control capability. This type of controller is illustrated in Figure 4.3. It functions analogously to the integral term in a PID (proportional-integral-derivative) controller where the integral term minimizes long term errors.

An important application note for dual controllers is that if feedback is used, then the feedback signal should be sent directly to the neural network instead of providing the network with an error signal. The error signal does not contain the context information of the feedback data. Thus, with the error

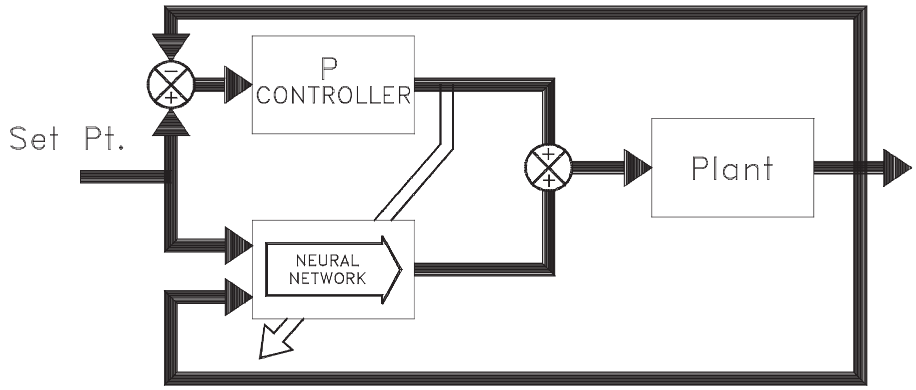

Figure 4.3: Combining a proportional controller with a neural network.

signal as input, the network is forced to find a mapping without knowledge of where in the state space the error is generated.

Kawato *et al.*, (1987) and Miyamoto *et al.*, (1988) successfully used this method to train a functional network to control a robotic arm. Hoptroff *et al.*, (1990) also experimented with dual control of a robotics arm, but found local minima to be a problem.

4.4 Inverse Control

Inverse control was studied as early as 1978 by Widrow *et al.*, (1978). The network controller is trained by learning the inverse of the plant. This is illustrated in Figure 4.4. The resulting network can be used directly as a controller since it has learned the necessary mapping to provide control. The

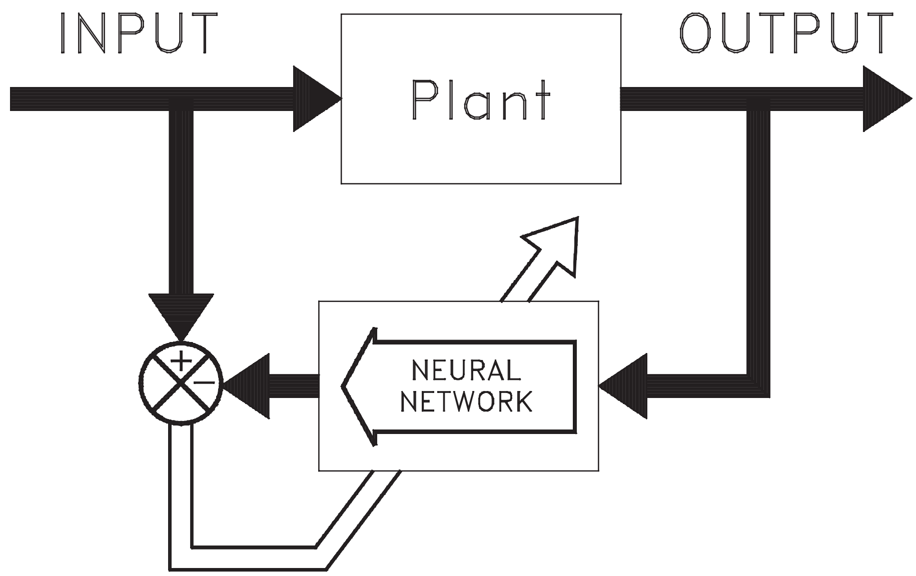

Figure 4.4: Training a neural network for inverse control.

use of an inverse trained neural network as a controller is illustrated in Figure 4.5.

A major problem with this type of control is the one-to-many problem. Many plant inputs may produce the same output. Thus the network will try to map a given plant output to many different inputs. The result is a mapping of the average over many targets. Usually the average is not the inverse, but for systems where the inverse is correct this is a good technique.

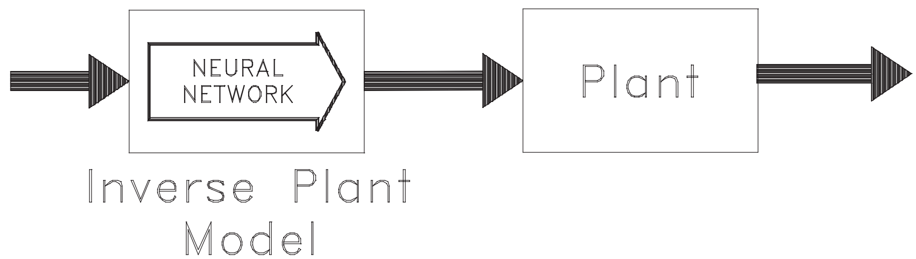

Figure 4.5: Utilizing an inverse trained network as a controller.

4.5 Plant Model Controllers

A forward model of the plant is free from the one-to-many problem, and it can be used to design a controller. First, a neural network, Figure 4.6, is used to identify the system model.

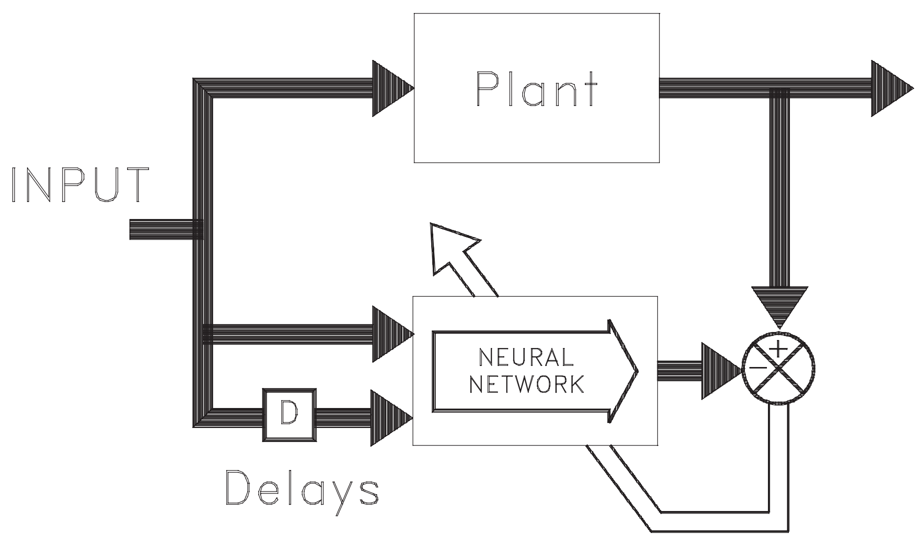

Figure 4.6: System identification with a neural network (Delay input is optional)

Once the neural network is trained, error can be back-propagated through the neural network plant model to train a neural network controller. The error, however, is different from the error used to train the system identification network. The system identification network is trained by using the difference between the network output and plant output, while the controller network is trained with the error between the plant output and the desired output. Figure 4.7 illustrates the way in which error is back-propagated through the model network to train the controller network.

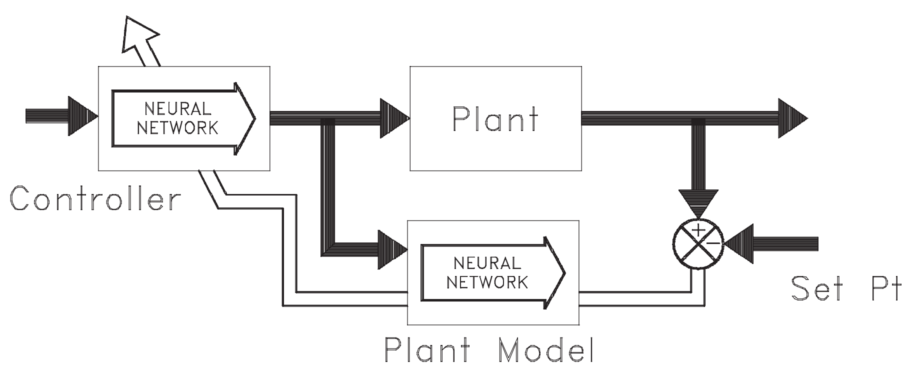

Figure 4.7: Back-propagating error through a system model neural network to adapt a control neural network.

There are many variations and application successes of neural network model systems (Narendra and Pathasarthy, 1990; Narendra and Pathasarthy, 1991; Barto, 1989; Jordan, 1988; Jordan and Jacobs, 1990a; Jordan and Rumelhart, 1990b; Hudson *et al.*, 1990; McAvoy *et al.*, 1989;

Werbos, 1990c). An important issue in all the studies is how to represent time. This is necessary since the plant model is used to compensate for the phase difference across the actual system. Thus, the plant model is often a predictor of the effect of current control inputs on later system response.

If the desired output is only available after a series of steps (*e.g.*, backing up a truck tractor and trailer), back-propagation through time (BTT) may be used (Nguyen and Widrow, 1989; Werbos, 1990c; Jordan and Jacobs, 1990a; Jordan and Rumelhart, 1990b). This is known as learning with a distal teacher. BTT is accomplished by recursively applying the controller and model network and saving both networks' states. Once the teacher is available, the error is first applied to the immediate state. Then it is propagated backward through both the model and controller networks to the previous time step. This is continued recursively until the beginning of the time series.

4.6 Reinforcement Learning Control

A reinforcement controller is trained with an error signal that is much lower in quality than the feedback signal in other adaptive controllers. The advantage is that the error signal is normally directly available from the environment with little or no processing.

A reinforcement learning system requires two components: a search generator and a memory. The search element is necessary to explore the space of operation and elicit responses from the environment. The memory component uses the reinforcement signal to learn the operations that yield the best return.

Often a reinforcement signal is not available at every instance of time. Thus, the learning system must learn a proper series of steps instead of a single step. To speed learning when no reinforcement signal is present, Barto *et al*., (1983) included an additional element, an adaptive critic (Figure 4.8). The purpose of the critic is to predict the reinforcement given the current system state. The critic provides a reinforcement signal when the environment is not providing one. This greatly speeds the learning.

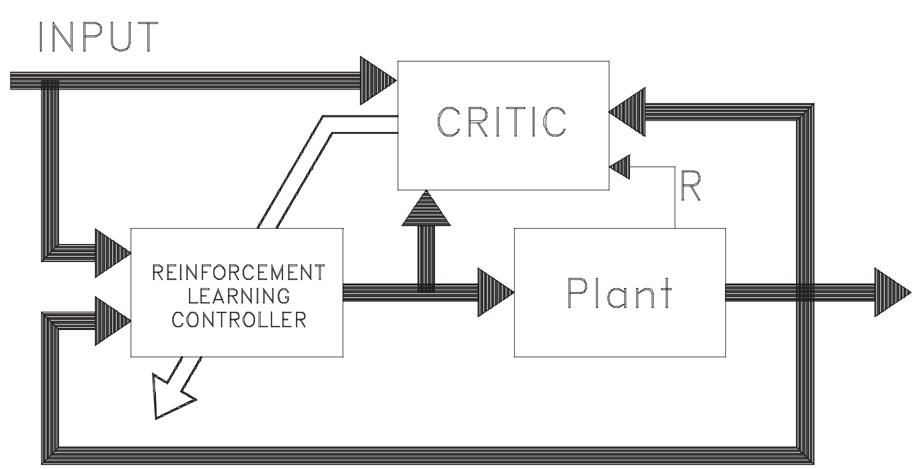

Figure 4.8: Reinforcement learning controller with adaptive critic

A difficulty with reinforcement learning is that because the reinforcement signal is only a single good/bad signal, the control network learns slowly in high dimensional spaces. Werbos (1990b) posits a solution to this problem by adding a model network between the critic and the controller (Figure 4.9). The model network provides derivative information to the controller network to speed the learning.

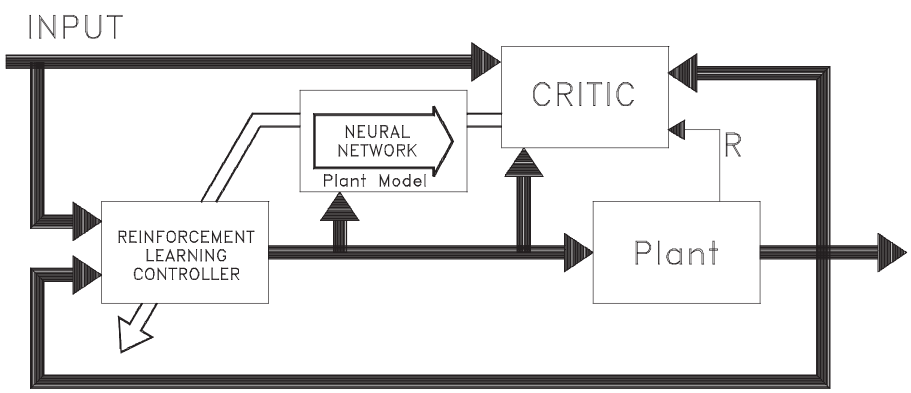

Figure 4.9: Reinforcement controller with plant model assisting the adaptive critic.

According to Werbos (1990b), BehavHeuristics of College Park, Maryland and McDonnell-Douglas have had success with the this type of learning system. Jameson (1990; 1991), however, reports problems with local minima.

The other facet of reinforcement learning is the search element. The search element is responsible for exploring the state space. It generates the test actions for probing the system. Much of the work to date has used random behavior as the search element. Recently, however, Schmidhuber (1990b) has proposed using a plant model for guiding the search.

Details aside, the importance of the reinforcement learning controllers is that they actively participate and experiment with the environment instead of being trained off-line. As such they have a greater capacity for learning about and accommodating unforeseen events in the environment than do other neural control techniques.

4.7 Optimal Neural Network Control

In the control literature, optimal control refers to control that minimizes a predetermined objective function. Since the foundation of a back-propagation neural network is nonlinear optimization, any of the above controllers could become optimizing by simply augmenting their existing objective functions. Plant model and reinforcement learning are most amenable to use as optimal controllers.

Plant model controllers are generally used as optimal controllers. The model neural network is trained independently of the controller neural network. By simply changing the controller network's objective function, the controller becomes an optimal controller. Reinforcement controllers are similar since only the critic's objective function needs to be modified.

Plant model neural network controllers as presently implemented often find the local minimum of the plant. In other words, given the current state of the plant, the optimal plant model controller can determine the best control settings to optimize that condition. To optimize systems over time or to search the plant's operational domain, a reinforcement controller is more appropriate.

Six techniques for neural network control were presented. Each applies a different solution to the distal-error problem. The neural control techniques provide a foundation to understand the proposed control architecture. The relationship between neural network control and optimal control was also presented.