STOCHASTIC OPTIMIZATION OF NEURAL NETWORKS AND IMPLICATIONS FOR BIOLOGICAL LEARNING

by

RUSSELL WAYNE ANDERSON

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

BIOENGINEERING

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA

San Francisco

[redacted]

Date University Librarian

Degree Conferred: 9/8/91

Stochastic Optimization of Neural Networks and Implications to Biological Learning

Dissertation Committee:
Hans J. Bremermann (Chair), Vojtech Licko, Donald A. Glaser,
Steven L. Lehman and V. (Rao) Vemuri

To Jennifer, my wife, best friend and companion, for listening.

ACKNOWLEDGEMENTS

I wish to thank my parents, Dr. Allen and Renee Anderson, and my family, who believed in me and supported me.

I would also like to express my appreciation for the people who most influenced my interests and development: Mr. Gonzoles, my English and Spanish teacher, who started me writing; Richard Thorn, my high school economics teacher, who first introduced me to mathematical models; Mike Baxter, my high school English teacher, who taught me the joy of reading; Dr. Edwin R. Lewis, who introduced me to mathematical biology and who impressed on me the value of scholarship in research; Dr. Lawrence Stark for his enthusiasm and support; Dr. Harlan Romberg, who exposed me to decision theory and taught me the meaning of the expression: "Polishing a turd"; Dr. Rajendra Bhatnagar for his friendship and advice; Dr. Vojtech Licko, for our many conversations, and who taught me a respect for rigor; Steven L. Lehman, for his technical advice and friendship; Dr. Farid U. Dowla, for his financial support and judgement; Dr. V. (Rao) Vemuri, for his advice, support and insights on work and living; Dr. Donald A. Glaser, for his advice and interest; and my many friends, for their continuing support.

Finally, special thanks goes to my research advisor and dear friend, Dr. Hans J. Bremermann. He helped me develop my ideas, exposed me to the history and state of the art of mathematical biology, and convinced me to do work which I found hard or tedious - all with the patience of a true master. He always had an eye out for The Big Picture, and through hundreds of stream-of-consciousness discussions, he shaped my understanding of the world.

This work was supported in part by NIH Training Grant # GM 08155 awarded to U.C. San Francisco; the U.S. Department of Energy (at Lawrence Livermore National Laboratory [LLNL]), contract W-7405-ENG-48; a grant from the Institute for Scientific Computing Research, LLNL awarded to U.C. Davis; and NIH Training Grant #GM-07379 awarded to U.C. Berkeley. Many of the computer simulations were done on the facilities at LLNL and the Department of Applied Sciences, U.C. Davis at Livermore.

Additional financial support for my graduate education and other research was provided by my parents, Dr. and Mrs. Allen M. Anderson, a Teaching Assistantship in Physiological Optics 129 (under Dr. Lawrence Stark, School of Optometry, U.C.B), a Teaching Assistantship in Electrical Engineering 100 (under Dr. Selig Kaplan, Dept. Nuclear Engineering, U.C.B), and NIH Biomedical Research Support Grant RR07006-24 at U.C. Berkeley.

Russell Wayne Anderson
Berkeley, California
August, 1991

Stochastic Optimization of Neural Networks
and Implications to Biological Learning

Russell Wayne Anderson

ABSTRACT

Neural network training algorithms, such as back-propagation, have been used in a wide variety of non-trivial applications, but are criticized as being biologically implausible. Hebbian learning rules can be supported biologically, but are insufficient for learning complex mappings. We propose a neurobiologically feasible rule for adjusting synaptic weights (called the "chemotaxis algorithm") which can produce the same results as back-propagation. Training consists of a biased Gaussian random-walk in weight space: synaptic modifications are generated locally, via random processes, and are retained only if a global reinforcement signal is broadcast to the network. We present theoretical arguments, biological evidence and computer simulations to support this learning rule.

The chemotaxis algorithm is demonstrated on benchmark problems, such as n-bit parity and the "encoder" problem. Theoretical capabilities neural networks are discussed and exploited in a real-world application in seismic signal classification. For this application, neural networks are competitive with conventional statistical pattern recognition methods.

The chemotaxis algorithm does not require any modifications for novel activation functions or recurrent networks, in this sense, it is a *general* learning rule. However, we demonstrate that training recurrent networks to perform complex temporal signal processing tasks is extremely slow and inefficient.

Finally, we develop a neural network methodology for motor control. Feed-forward networks are used to estimate the parameters of a temporal control signal. State trajectory error is then used for network performance evaluation. Since the chemotaxis algorithm does not require the calculation of gradients of unknown state variables, it offers unique advantages over the back-propagation algorithm for this application. This methodology is applied to a linear, second-order system and a two-link, planar manipulator. Analogies to motor control via central pattern generators and the acquisition of skilled movement are presented.

TABLE OF CONTENTS

PROLOG

General Philosophy. 1

Genesis of a Learning Algorithm 2

Motivation for the Selected Applications 4

CHAPTER 1: INTRODUCTION: NEURAL NETWORK MODELING

1.1 Neural Networks are Universal Computational Devices. . 7

1.2 Neuronal Networks Employ Similar Structures for Diverse Functions 9

1.3 Hebbian Learning Rules 9

1.4 Theoretical Learning Rules 11

1.5 The Success of Back-Propagation 12

1.6 Neural Network Models of Biological Systems 13

1.7 A Biologically Plausible Learning Rule 15

CHAPTER 2: A BIOLOGICALLY PLAUSIBLE LEARNING RULE

2.1 Introduction 18

2.2 Training for Performance 19
- 2.2.1 Error Function and Back-Propagation 19
- 2.2.2 Search Algorithms 22

2.2.3 Mutational Versus Gaussian Random Walks 23
2.2.4 Importance of the Search Probability Distribution 25
2.2.5 Learning the Step Size 27
2.2.6 The Chemotaxis Algorithm 28
2.2.7 Accelerated Guided Random Search 33

2.3 Random Walk Learning in Biological Neural Systems . . . 34
2.3.1 Gaussian Fluctuations of Neuronal Connections and Synaptic Strength 34
2.3.2 Fixation of Successful Modifications 35
2.3.3 Origin of the Global Signal 36

2.4 Simulation Examples: Training Networks on Benchmark Problems 37
2.4.1 N-Bit Parity 37
2.4.2 Discussion 38
2.4.3 The 4-Bit Encoder Problem 40
2.4.4 The Multilayer Encoder 42

2.5 Variations of the Basic Algorithm 42
2.5.1 Boltzmann Factor 42
2.5.2 Activation Functions 44
2.5.3 Net Expansion and Contraction 45

2.6 A Conjecture for Cortical Mappings 46

2.7 Discussion: Observability of Biological Learning 49

CHAPTER 3: NEURAL NETWORK THEORY

3.1 Introduction 53

3.2 Neural Networks and Function Approximation 54

3.3 Pattern Recognition 59
3.3.1 Feature Extraction 60
3.3.2 A Brief Review of Statistical Decision Theory . . . 61
3.3.2.1 The Two-Category Case 62
3.3.2.2 Discriminant Functions 64
3.3.3 Nonparametric (Distribution-Free) Classification. 66
3.3.4 Neural Networks Can be Used to Approximate the Bayes-Optimal Discriminant Function 67
3.3.5 Advantages of the Neural Network Approach . . 68

3.4 Generalization 69
3.4.1 Boolean Functions 69
3.4.2 Limitations of Generalization 70
3.4.3 A Simulation Example 72
3.4.4 Discussion: Generalization, Cortical Maps and Learning Rules 77

CHAPTER 4: AN EXAMPLE OF PATTERN DISCRIMINATION WITH NEURAL NETWORKS ON SEISMIC DATA

4.1 Motivation 78
4.1.1 The Real World of Pattern Recognition 78
4.1.2 Neural Networks versus Statistical Methods for Discrimination of Seismic Signals 79
4.1.3 An Engineering Application of a Biologically Inspired Computational Method 80

4.2 Background: Seismic Signal Discrimination 80
4.2.1 Nuclear Test Ban Treaties 80
4.2.2 Characteristics of Seismic Data 81
4.2.3 Earthquakes versus Explosions 81
4.2.4 Discrimination Studies 83

4.3 A Neural Network Approach 85

4.4 Seismic Spectral Discriminants 88
4.4.1 Characteristics of the Spectral Data 88
4.4.2 Seismic Discriminant Functions 91

4.5 Seismic Database and Preprocessing 93
4.5.1 Spectral Estimation and Signal-to-Noise Ratio Checking 95
4.5.2 Correction of Distance Effects on the Spectra . . . 97
4.5.3 Normalization of Spectral Levels for Magnitude Invariance 99

4.6 Neural Network Discrimination Performance 101
4.6.1 Training with Chemotaxis and Back-Propagation 101
4.6.2 "Leave-One-Out" Performance 102
4.6.3 Discussion 105

4.7 Comparison with the Fisher Discriminant 105

4.8 Conclusions 108
4.8.1 Advantages over Conventional Methods 111

4.9 Subsequent Work on Seismic Signal Discrimination 112

CHAPTER 5: RECURRENT NETWORKS AND THE CHEMOTAXIS ALGORITHM

5.1 Introduction 113

5.2 Recurrent Network Theory 114

5.3 Training Algorithms for Recurrent Networks 115

5.4 Modifications of the Chemotaxis Algorithm for Recurrent Networks 117

5.4.1 Network Architecture 117
5.4.2 Definition of the Objective Function 119

5.5 Simulation Results 120
5.5.1 A Neural 'Integrator' 120
5.5.2 Training a Recurrent Network to be a Finite-State Machine 123
5.5.3 Oscillators 128

5.6 Discussion 131
5.5.1 Alternatives to Recurrent Neural Networks in Control 132

CHAPTER 6: NEURAL NETWORKS APPLIED TO OPEN-LOOP, DYNAMIC CONTROL

6.1 Introduction 134
6.1.1 Parameterization of Temporal Sequences in Neural Network Applications 135
6.1.2 Parameterization of Time-Optimal Control Signals 135
6.1.3 A Neural Network Control Methodology 136

6.2 Time-Optimal Control Theory 138
6.2.1 Problem Statement 138
6.2.2 Pontryagin's Maximum Principle 138
6.2.3 Bang-Bang Control 139

6.3 Example: Time-Optimal Control of a Linear Second-Order System 141
6.3.1 Calculation of the Switching Function 143
6.3.2 Optimal Control with a Single Switching Time . . 146
6.3.3 Switching Times as a Function of System Parameters 149

6.4 Neural Networks for Open-Loop Control 149
6.4.1 Training a Neural Network to Generate Switching Times 149
6.4.1.1 Experiment #1 151
6.4.1.2 Experiment #2 153
6.4.1.3 Experiment #3 155
6.4.1.4 Conclusion 155
6.4.2 Training Neural Networks to Control a System with Unknown Switching Times 158
6.4.2.1 Simulation Results with a Second-Order Plant 161
6.4.2.2 Discussion 162
6.4.3 Preliminary Results with a Multilink Arm 167
6.4.3.1 Dynamics of a Planar Manipulator 167
6.4.3.2 Preliminary Results 170

6.5 Discussion 172
6.5.1 Future Outlook 172
6.5.2 Relation to Biological Control 173

REFERENCES 174

LIST OF TABLES

2.1 Comparison of the chemotaxis algorithm and back-propagation 39
2.2 Performance of the chemotaxis algorithm on the encoder problem with communication "bottlenecks". 41
2.3 Comparison of training times for the multilayer encoder problem 43

3.1 Training sets used in the generalization study. 73
3.2 Generalization as a function of training set and net architecture for the encoder problem. 76

4.1 Performance of the neural network 103
4.2 Performance of the Fisher discriminant 107

5.1 Average training times for sequential parity 127

6.1 Summary of results for section 3.1 157
6.2 Summary of results for section 3.3 163

LIST OF FIGURES

2.1 Uniform versus Gaussian distribution of random directions 26
2.2 Graphical presentation of the random walk in weight space 30-32
2.3 Schematic representation of the map from the retina to the lateral geniculate nucleus. 48

3.1 Schematic representation of a neural network applied to function approximation 57
3.2 Example of function approximation 58
3.3 Probability densities of two object classes on feature x . 63
3.4 Distribution of two object classes on two features 65
3.5 4-bit Hamming space 74

4.1 Examples of peak-to-peak earth-motion amplitude measurements for an earthquake and an explosion. 82
4.2 Network-averaged discriminants. 84
4.3 The architecture of a neural net used to discriminate between explosion and earthquake spectra. 87
4.4 Examples of seismograms for (a) an earthquake and (b) an underground nuclear explosion. 89
4.5 Examples of displacement amplitude spectra of the P_n, P_g and L_g phases. 90
4.6 Map of the LLNL Nevada Test Site (NTS) seismic network 94
4.7 Histogram of event distances. 96
4.8 Spectral correction function. 98
4.9 Magnitude distribution of the two populations. 100
4.10 a-b Plots of the normalized (a) L_g and (b) P_g spectra. . 104
4.11 The weights from the input to a hidden unit for a trained network at Elko. 109

4.12 The weights corresponding to the P_g spectra for the other three stations. 110

5.1 Architecture of a recurrent neural network. 118
5.2a Performance of a 'neural integrator' on the training set.121
5.2b Performance of a 'neural integrator' on the test set. . . . 122
5.3 State transition matrices for a two-state FSM 124
5.4 Training a recurrent neural network to be a finite state machine. 126
5.5 Performance of neural networks trained to oscillate. . . . 129
5.6 Implementation of a sinusoidal oscillator with a recurrent network. 130

6.1 Schematic representation of a problem in open-loop motor control 137
6.2 Time optimal control of a second-order, linear system. 145
6.3 Optimal switching and stopping times. 148
6.4 Schematic representation of three experiments. 150
6.5 Performance of a trained network in experiment #1. . . 152
6.6 Network performance for experiment #2. 154
6.7 Network performance for experiment #3. 156
6.8 Experiment design used in section 3.2 159
6.9 Evaluation of system performance 160
6.10 System performance for experiment #1. 164
6.11 System performance for experiment #2. 165
6.12 System performance for experiment #3. 166
6.13 Geometry of a two-link, planar manipulator 168
6.14 Arm trajectory generated by a neural network 171

STOCHASTIC OPTIMIZATION OF NEURAL NETWORKS AND IMPLICATIONS FOR BIOLOGICAL LEARNING

PROLOG

GENERAL PHILOSOPHY

The main result of this thesis is the development and exploration of a biologically plausible learning rule. By necessity, this study involves elements of several normally distinct fields - including neurobiology, mathematics, computer science, complexity theory, pattern recognition and control theory.

Plasticity in neuronal networks is observable macroscopically (in the changes in cortical maps) and microscopically (at individual synaptic connections). However, the mechanisms that govern learning are still very much in dispute and virtually unobservable with present day methods. Alternatively, many groups have studied neural network models that imitate the connectivity and properties of real, biological neuronal structures. Neural network models can be used to investigate the capabilities of these ensembles and to test learning mechanisms. The rules that have emerged from such studies, so far, are either insufficient for complex learning tasks or have been criticized as being biologically implausible.

Neural network models, to some extent, can be described and explored mathematically. Analysis of a network consisting of a handful of neurons, however, can become mathematically intractable due to the large number of variables and the complexity and nonlinearity of their interactions. Most purely mathematical treatments of neural networks have been limited to existence proofs; in other words, they tell us what these structures can do *in principle.* (These results are discussed in detail in Chapters 3 and 5.) To discover what neural networks can do *in practice*, and to find what learning rules can find these solutions, we must rely on computer simulations of properly chosen learning tasks.

Much of the data processing in the brain is involved in pattern discrimination and motor control. And since the neural structures involved in both sensory and motor functions are histologically very similar, we would like any potential learning rule to be robust enough to solve both pattern recognition and motor control problems.

In the course of this thesis, we train neural networks for widely varying tasks: implementation of Boolean functions, encoding/decoding of topographic maps, pattern recognition and motor control. We note that through the experience of creating successful neural network applications we may gain insight on how actual neural structures solve similar problems.

GENESIS OF A LEARNING ALGORITHM

I began working with Dr. Bremermann in 1988, having been inspired by his seminar series on "Brain Theory". Initially, we had a modest research project in mind. We wished to apply an optimization algorithm developed by Dr. Bremermann to neural network training.

The task of learning in neural networks is to find a vector in parameter space (weight space) which satisfies a goal defined by a performance criterion, or objective function. The most common neural network training algorithm, back-propagation of error, is an iterative, gradient descent optimization algorithm. The Bremermann optimizer searches parameter space by estimating the objective function along random rays with Lagrangian interpolation (Bremermann 1970). The optimizer can be quite efficient as long as the error surface of the objective function (also called the "energy" or error surface) has the same general shape as a the polynomial used for an estimate (Milstein 1975, Lehman and Stark 1979, Subba Rao 1986, Cutler 1988). We had hoped that this algorithm would be faster than gradient descent methods and have the added advantage of being able to escape local minima through long-range interpolation.

Regrettably, Lagrangian interpolation turned out to be a very poor predictor of neural network error surfaces. We noticed, however, that along the random rays we were able to locate lower energy points without the aid of interpolation. We decided to explore training without the interpolation portion of the optimizer.

What remained was a training algorithm which consisted of a biased Gaussian random-walk through weight space. We have dubbed this algorithm "the chemotaxis algorithm" since it is mathematically analogous to the way bacteria find higher concentrations of food (Berg 1975). (Dr. Bremermann suggested "NETWALK" in a pun on Sejnowski and Rosenberg's famous neural network paper called "NetTalk" [Sejnowski and Rosenberg 1987]. The reader can decide what would have been the better choice.)

Optimization through random walks is nothing has not heretofore been vigorously tested on neural networks. There had been some experimentation with a similar algorithm in the late 1960's (see section 2.2.7); however, the results obtained then (hampered by the lack of ample computer power) and failure to apply this method to non-trivial tasks resulted in a lack of follow-up, and relegated this method to relative obscurity. Also overlooked by researchers at the time was that a biased random walk is a biologically plausible learning rule. We present mathematical arguments, biological evidence and computer simulations to support this learning rule.

MOTIVATION FOR THE SELECTED APPLICATIONS

In the course of this thesis, we demonstrate the effectiveness of the chemotaxis algorithm on several abstract, benchmark neural network problems as well as on non-trivial, real-world problems in pattern recognition and motor control. In the former category we train feed-forward networks to solve the N-bit parity and encoder problems. In addition, we train fully-connected networks to integrate a sequential signal, mimic a simple finite-state machine and oscillate at various frequencies.

In Chapter 4, we tackle a practical problem in pattern recognition, and in Chapter 6, we develop a neural network methodology for motor control. The specific pattern recognition problem we chose is discrimination of seismic signals generated by earthquakes and underground nuclear explosions. It is a "real-world" problem, which was investigated in a non-classified setting at Lawrence Livermore National Laboratory (LLNL). Neural network methodologies were competing with more conventional pattern recognition methods based on feature extraction and statistical decision theory. In this setting, we could test the efficacy of the chemotaxis algorithm versus commercial software for neural network training. The choice of applications was also guided by pragmatic reasons: opportunity, access to real data, the potential for collaboration with experts in the field, computer resources, funding opportunities, personal interests, inspiration and serendipity.

In December 1988 I was given a unique opportunity to study the usefulness of neural networks in a practical problem in pattern recognition when I met and was later hired by Dr. Farid Dowla at LLNL. Dr. Dowla and I shared a mutual interest in neural networks and a background in decision theory problems. I had worked on threat discrimination while employed by the Ballistic Missile Defense Program (McDonnell Douglas Astronautics, Huntington Beach, CA), and Dr. Dowla worked on seismic signal discrimination for the Treaty

Verification Program (Earth Sciences Division, LLNL). The problem of discriminating re-entry vehicles from booster tank fragments and decoys is well-characterized with discriminants tightly coupled to the *underlying physics* of projectile motion. The cause and characteristics of seismic waveforms, in contrast, is not well understood. The seismic database is inherently high-dimensional, requiring extensive pre-processing to reduce the problem to a manageable size. Many of the most effective discriminants used have been found empirically or through intelligent guesswork. The problem seemed to us to be ideally suited to a neural network application.

Dowla and others (Dowla et al. 1989) had already attempted pre-processing temporal seismic signals using neural networks and an in-house transputer. This study encountered problems with the so-called *time registration problem*, where small shifts in the alignment of the input window would invalidate the results. We discussed using instead the Fourier transform of the input signal to avoid windowing problems, and I was eager to try the chemotaxis algorithm on some real problems. The results of our studies are presented in Chapter 4.

Pattern discrimination ability is a form of knowledge. This knowledge seems to be stored in the organization of the neural structures which process this information (e.g. the retina, lateral geniculate nucleus and visual cortex). Likewise, skilled motor control is a form of "knowledge" stored in the motor cortex, cerebellum and spinal cord. For this reason, I was next interested in applying neural networks to motor or effector functions.

I had previously studied the control of extraocular muscles involved in saccadic eye movements with Dr. Lawrence Stark (and worked as a teaching assistant in Physiological Optics). Lurking in the back of my mind for several years how the neural structures computed the control signals for driving this open-loop system. As I experimented with recurrent networks, I began to realize that "teaching" a randomly connected group of neurons to generate these control signals was extremely difficult (see Chapter 5).

In Chapter 6, we suggest alternative neural network strategies to generate temporal signals for motor control. The particular application chosen is open-loop motor control. We present examples of time-optimal control of a second-order linear system and a multilink robotic arm.

CHAPTER 1

INTRODUCTION: NEURAL NETWORK MODELING

Much of the 'knowledge' stored in the nervous system is thought to be imbedded in the structure, or 'wiring diagrams' of assemblies of neurons. Many neuronal assemblies have clear, identified functions (e.g. the stretch reflex in the spinal cord, control of the stomach muscles by the lobster stomatogastric ganglion [Miller and Selverston 1982], motion detection in the retina [Poggio and Koch 1987]). Many of these systems are generally thought of as fixed, or "programmed from birth". But there is a great deal of plasticity in the connectivity, cortical representation and organization of neural structures, and this ability to change continues throughout life (Merzenich et al. 1988). The two main issues addressed in this thesis are:

(1) What capabilities do these adaptive structures confer to the organism? , and

(2) What processes govern this change?

1.1 NEURAL NETWORKS ARE UNIVERSAL COMPUTATIONAL DEVICES

Mathematical analyses of neural networks reveal them to be potentially powerful computing devices. In their landmark paper, *A Logical Calculus of the Ideas Immanent in Nervous Activity*, McCulloch and Pitts (1943) demonstrated how a network of extremely simplified ("all-or-nothing") neurons could compute any Boolean function and therefore could emulate any Turing machine in the following sense:

> "...every net, if furnished with a tape, scanners connected to afferents, and suitable efferents to perform the necessary motor-operations, can compute only such numbers as can a Turing machine... and that nets with circles [feedback loops] can compute, without scanners and a tape, some of the numbers the machine can, but no others and not all of them."

J.D. Cowan comments:

> "This was a remarkable achievement. It established once and for all, the validity of making formal models of brain mechanisms, if not their veridicality."(Cowan 1990)

In the forties and fifties, mathematicians and logicians were concerned with characterizing classes of functions that were computable by *ideal* Turing machines (without concern to the time such computations would take). McCulloch and Pitts offered a solution to the "embodiment problem" by showing how networks of biological neurons could perform the same calculations as a Turing machine. Rosenblatt (1958) demonstrated that these networks, when endowed with modifiable connections ("perceptrons"), could be "trained" to classify patterns (see also Cowan 1990, Arbib 1987). Thus, Rosenblatt was the first who bridged the gap between behaviorism and modular nets of McCulloch and Pitts neurons.

Rosenblatt went on to explore quantitatively what problems his perceptrons could solve, how many "neurons" would be required, and how long they had to be trained (Rosenblatt 1962). Beginning in the sixties, theoretical computer scientists had become increasingly concerned about computational complexity (Bledsoe 1961c, Bremermann 1962, Beckenstein and Schiffer 1990). The best algorithm or computational method for a given task is known only in a few elementary cases (Bremermann 1974b). Hence it becomes important to test performance of alternative computational methods (in this case, learning rules) empirically.

Since then, a variety of neural network models and training algorithms have been developed to tap this potential computing power (for a review, see Lippmann 1987). These mathematical models allow us to study the interactions of many adaptive neurons and to test hypothetical learning mechanisms.

1.2 NEURONAL NETWORKS EMPLOY SIMILAR STRUCTURES FOR DIVERSE FUNCTIONS

Assemblies of neurons in the brain are involved in widely varied functions (vision, sensory perception, motor control, etc.), yet are histologically very similar. In fact, it takes a highly trained specialist to distinguish between neurons taken from sensory and motor cortex microscopically (or even between brain tissue from different *species*! [Braitenberg, personal communication]). These histological similarities between sensory and motor neurons indicate that general organizational principles may be involved in brain function.

In this work, we employ neural network models to many functionally different applications. We show that a single learning rule can train neural network models to perform nonlinear mappings (Chapter 2), pattern classification (Chapter 4) and motor control (Chapter 6).

1.3 HEBBIAN LEARNING RULES

In 1949, Hebb proposed a neuronal learning rule which could integrate associative memories into neural networks (Hebb 1949). Hebb postulated that when one neuron repeatedly excites another, the synaptic knobs are strengthened. Verification has taken time, but there is now ample evidence that Hebbian-type long term potentiation (LTP) (with some modifications of the original hypothesis) does indeed occur (Lynch 1986, Kennedy 1988, Stevens

1989). Long-term synaptic change in *Aplysia* sensorimotor synapses can be produced by a cellular analog of classical conditioning (Buonomano and Byrne 1990). Long-term depression (LTD) has been observed in the same system supporting a corollary "Hebbian covariance learning rule" (Stanton and Sejnowski 1989).

Theoretically, Hebbian learning can account for some types of biological learning. For example, a Hebbian mechanism has been shown to be sufficient to account for plasticity in mappings of cortical representation (Grajski and Merzenich 1990). But there is more to the brain than conditioned reflexes and associative memories. For anything but special cases, Hebb's rule is insufficient as a learning rule (Rosenblatt 1962; Rumelhart and McClelland 1986).

Since Hebbian learning requires near simultaneous or synchronous stimuli, it is limited *temporally.* In many real biological situations, instantaneous performance results are not available. Hebbian learning would have to be combined with some type of memory mechanism to account for many phenomena observed in psychophysical and electrophysiological studies of classical conditioning - such as anticipation of the unconditioned stimulus (Deno 1991, Chester 1990b). Recent attempts to expand Hebbian learning rules to include short-term memory (Sutton and Barto 1983, Klopf 1989, Grossberg and Schmajuk 1989) have met with limited success (Chester 1990b).

Since the Hebbian rule applies only to correlations at the synaptic level, they are limited *locally.* Strengthening a local correlation in the context of a nonlinear mapping of several variables (such as the N-bit parity problem) often reduces overall performance. Rumelhart and McClelland (1986) demonstrated that a Hebbian learning was rule unable to train a multilayer perceptron network to learn one of the simplest nonlinear mappings (the "exclusive OR" problem). The task of determining the relative responsibility of individual synaptic changes in a network is referred to as the "credit assignment problem".

1.4 THEORETICAL LEARNING RULES

There is no general method for *finding* learning rules, but once a particular one has been proposed, one can analyze it by proving mathematically that it converges (and how fast), or one can explore it through computer simulation or hardware implementation.

In 1956 Frank Rosenblatt formulated his "Perceptron" model and set out to explore all three approaches. Analytically, he described an iterative error correction procedure called the "perceptron learning rule". He also proved that this procedure will converge if there is a solution to a classification problem within the constraints of the given neural net. Today this algorithm for changing synaptic weights is also known as the "Widrow-Hoff rule" (Widrow and Hoff 1960). It can be proven to converge, it is reasonably efficient, but unfortunately it works only for "*simple perceptrons*", which in today's terminology are equivalent to neural nets without "hidden units". This limitation, acknowledged by Rosenblatt, helped kill perceptron research in the late 1960's.

From the beginning, Rosenblatt was interested in exploring "multilayer perceptrons". The input-response function of such nets is inherently nonlinear and mathematically intractable. Neither the "error correction procedure" nor Hebb's rule are a viable training procedure for these networks. The problem *can* be solved, however, by numerical minimization of the "squared error performance function".

There are several engineering techniques available to optimize a system with a defined error function, among them "genetic" and gradient descent algorithms. The genetic approach is to treat the synaptic weights like nucleotides of DNA; mutate, recombine, cross over and select, as in Darwinian evolution. Early on, Bremermann and Bledsoe experimented with the idea of applying genetic algorithms to optimizing the performance of perceptrons (Bremermann 1958, Bledsoe 1961a,b, Bremermann 1962, Bremermann and Rogson 1964, Bremermann, Rogson and Salaff 1966). The method works *in principle* (and is currently a popular method in Artificial Intelligence [Holland 1975]). But success with neural networks did not come until much later due to the extraordinary computational requirements (Edelman 1987, Montana and Davis 1989, Austin 1990). It took a surprisingly long time for an effective gradient descent algorithm to be developed.

1.5 THE SUCCESS OF BACK-PROPAGATION

More than a decade later Rumelhart, Hinton, McClelland, Sejnowski, and others (Rumelhart and McClelland 1986) had successes training multi-layer perceptrons. In order to develop a gradient descent algorithm, they replaced the discontinuous step function of the linear threshold elements (McCulloch and Pitts neurons) that Rosenblatt had employed by a continuous (and differentiable) activation function (Feldman 1981). The algorithm they developed is known as "back-propagation of error", because synaptic weight changes are determined by error gradients (partial derivatives) propagated back through the network via the chain rule. With steepest descent along gradients they were able to obtain some startling results: Pronunciation of English words (Sejnowski and Rosenberg 1987), elucidation of 3-Dimensional shape from 2-D shaded images (Lehky and Sejnowski 1988), predicting secondary structure in protein conformation (Qian and Sejnowski 1988), computing stereo disparity (Lehky and Sejnowski 1990), and selecting moves in "backgammon" (Tesauro and Sejnowski 1989). Surprisingly, moderate size nets performed some remarkable tasks. The nature of neural nets seems to make local minima less of a problem than they are in some other optimization problems of comparable size. (Two other applications are presented in chapters 4 and 6. One of these uses the back-propagation algorithm, the other does not - and as we shall see, cannot.)

1.6 NEURAL NETWORK MODELS OF BIOLOGICAL SYSTEMS

In 1962, Hubel and Wiesel studied the response of neurons in the cat's visual cortex to oriented bars of light (Hubel and Wiesel 1962; Hubel 1988). They found many of these neurons preferentially responded to specific orientations and that cells with similar receptive fields were organized spatially in the cortex. Hubel and Wiesel's neuron's came to be known as "edge detectors", because of their strong response to lines and boundaries. A conventional wisdom evolved which presumed that one could understand neural processing by identifying the receptive fields of individual neurons. This launched a decades-long effort in which neurobiology became dominated by the identification of cortical mappings and receptive fields through single electrode recording experiments.

Long neglected in these studies was the complexity of the dense connectivity present in the brain. Neural network modelers began to challenge the prevailing paradigm of neurobiology. Many likened this effort to "trying to understand the function of a computer by listening to the output of a single transistor". They stressed the *emergent* computational properties of *networks* of neurons (Hopfield 1984; Freeman 1991).

Hopfield and Tank (1986) provided a simple example of the capabilities of a very simple network of 7 neurons. When studied individually, they argued that each individual performed apparently random signal transformations; collectively, the network performed the operation of an A/D (analog-to-digital) converter. In their paper "Computing 3-D Curvatures from Images of Surfaces Using a Neural Model" (commonly referred to "Shape-from-Shading"), they dramatically illustrated the limitation of elucidating a neuron's *function* from a mapping of it's receptive field (Lehky and Sejnowski 1988). In their study, a two layer network was trained to identify the local curvatures of a shaded two-dimensional image. The images used in this study had no edges or boundaries. Yet, when they examined the hidden units, they found units with oriented receptive fields nearly identical to Hubel and Wiesel's "edge detectors". This does not rule out the possibility that edge detectors don't exist in the brain, but it illustrates the point that knowledge of an individual neuron's receptive field is insufficient to characterize its' function in the larger context of a network.

Neural network modelers have also tried to gain insight into the distributed nature of actual neural systems by training artificial neural networks on biologically realistic input/output relationships. Most efforts to 'bridge the gap' between artificial neural networks and electrophysiological recordings have been limited to 'static' or instantaneous pattern mappings. This approach has offered insights into neural structures which process *spatial* information, such as in the visual or somatosensory cortex (Lehky and Sejnowski 1988, Pearson et al. 1987, Grajski and Merzenich 1989).

Many studies have omitted temporal features of both the environment and the neural architecture while retaining essential features of the problem being addressed. However, pattern recognition in real cortical tissues typically involves transitory states of chaos and coherent oscillations (Freeman 1991, Baird 1990, Skarda and Freeman 1987). For example, refractory periods are an integral component in visual motion detection (Poggio and Koch 1987).

Many researchers have attempted to extend neural network methodologies to intrinsically temporal neural processes such as motor control (Anastasio et al. 1991, Lockery et al. 1990) or olfactory discrimination (Freeman 1991). Many of these studies employ dynamical (recurrent) neural network architectures. Lockery and colleagues have modeled the withdrawal reflex in the leech using static and dynamic neural networks to study the distributed function of interneurons in this well-characterized circuit (Lockery et al. 1988; 1990). In his methodology, he attempts to use whatever existing information about known neural relationships, receptive and projective, and uses neural network optimization algorithms to "fill in the missing pieces". Anastasio has conducted similar studies on the vestibulo-ocular reflex (VOR) (Anastasio et al. 1989; Anastasio et al., in press 1991). It is hoped that these models will one day provide a useful supplement to the traditional minimal implementation single-input/single-output (SISO) block diagrams used in biological control system models.

1.7 A BIOLOGICALLY PLAUSIBLE LEARNING RULE

Despite its many successes, neural network research has been plagued by one significant drawback. The most sophisticated neural network applications to date have trained networks using the back-propagation algorithm - or one of it's many variants. However, nobody has proposed a reasonable biological mechanism by which such a complicated algorithm could be implemented with identified biological structures. Computation of the gradient and adjustment of synaptic weights, individually through "back propagation of error", would require an error correction structure as complex as the "feed forward net" to propagate error signals back to the individual synapses. Also, the method becomes inefficient and incredibly complex in network architectures that allow lateral and feedback connections. Hinton, one of the pioneers of neural networks, acknowledges that "as a biological model, backpropagation is implausible" (Hinton 1989). Mel (1990, p 34.) asks,

> "[I]s it, then, a fundamental law that neural associative learning algorithms must be either representationally impoverished or mechanistically overcomplex?"

In his article "The recent excitement about neural networks", Francis Crick (1989) writes:

> "It is hardly surprising that such achievements [referring to the work of Rumelhart, Hinton, Sejnowski et al.] have produced a heady sense of euphoria. But is this what the brain actually does? Alas, the back-prop nets are unrealistic in almost every

respect....*Obviously what is really required is a brain-like algorithm which produces results of the same general character as back propagation*"[emphasis added].

In Chapter 2, we introduce such an algorithm: We have experimented with a method that is so simple and straightforward that it seems ideal for implementation in a biological structure. DNA can generate neural circuits through morphogenesis (though evolution would have taken many millions of years). Thus, *in principle*, DNA could have organized back-propagation networks in the brain. But, there is a trade-off between simplicity and whatever increased performance might be obtained from complex neural circuitry. The algorithm that we propose is of utmost simplicity. This is its merit. It does not require much "genetic information" or an excessive overhead of predetermined neural circuitry (as would back-propagation). It may not appeal to professionals who make optimization algorithms their life's work and who derive esthetic pleasure from the intricate program structures their algorithms require. However, whatever is programmed in the brain has to first be programmed in the genes. The conversion of nucleotide sequences in the genome into computationally competent neural nets is anything but simple. Economy in the number of genes involved and economy in developmental complexity of neural nets are therefore overriding powerful principles of evolutionary pattern formation and selection.

The algorithm that we are proposing, is a random walk through the space of synaptic weights. It relies on Gaussian trial fluctuations, which can be generated locally at the synapses, and which are then fixed by a global signal "To all synapses: Freeze the last trial fluctuation and make it the starting point for new trials". How can such a simple rule possibly be substitutable for back-propagation? We were able to show that this algorithm, augmented by a few additional features, is indeed a candidate for the synapse adjustment.

To obtain a comparison to other theoretical learning rules, we have applied our method to two benchmark problems: "n-bit parity" and the "encoder problem" (Chapter 2). In subsequent chapters, we present successful applications of Gaussian random walks and the "chemotaxis algorithm" to train nets to perform a variety of tasks: discrimination of earthquakes from underground explosions (Chapter 4), generation of time-varying signals from "recurrent networks" (Chapter 5), and the generation of control signals for open-loop, dynamical systems (Chapter 6). This is very encouraging since the essential point of Gaussian random walk descent is its utter simplicity in terms of biological implementation. By solving real-world problems with neural networks trained with a biologically plausible learning rule, we help bridge the gap between mathematical models and actual neuronal assemblies.

Perhaps such a learning rule could be in operation, at least in part, in some types of learning in the nervous system. The most likely candidates may be in the acquisition of skilled movement, refinement of perceptual skills, or the establishment and refinement of cortical mappings. In the following chapters, we present theoretical arguments, biological evidence and computer simulations to support this view.

CHAPTER 2

A BIOLOGICALLY PLAUSIBLE LEARNING RULE[1]

2.1 INTRODUCTION

In this chapter we describe a non-Hebbian learning rule for neural network training, called the "chemotaxis algorithm". Network training consists of a biased random walk through weight space, combined with a generalized reinforcement signal that fixes successful modifications. Section 2.2 is a discussion of search and engineering optimization algorithms, including the back-propagation and chemotaxis algorithms. The chemotaxis algorithm requires three elements: (1.) a source of random variation in synaptic efficacy, (2.) a structure for evaluation of network performance, and (3.) a reinforcement, or "print" signal to retain successful modifications. In section 2.3 we argue the biological plausibility of this learning rule by showing that these three elements are consistent with neurobiological structures and phenomena. In section 2.4 we demonstrate, using computer simulations, that this learning rule is sufficeint for learning the same complex, non-linear mappings as the back-propagation algorithm and is reasonably efficient when applied to benchmark problems in neural network research today. Section 2.5 presents variations of the chemotaxis algorithm. We show that our algorithm does not require modification for novel activation functions (such as step functions or conjunctive inputs) or for dynamical ('recurrent') networks; in this sense, it is a *general* learning rule.

---

[1] Much of the text and results presented in this chapter first appeared (or will appear) in publications by Hans J. Bremermann and RWA (Bremermann and Anderson 1990, 1991).

In the last two sections, we return to our discussion of biological neural networks. Our algorithm suggests a way in which cortical maps can be established, maintained and expanded. In Section 2.6 we offer a conjecture on cortical mappings based on these observations. We conclude with a discussion on the problems of observability of learning in biological systems and suggest an experiment for the identification of a reinforcement signal.

2.2 TRAINING FOR PERFORMANCE

2.2.1 ERROR FUNCTION AND BACK-PROPAGATION

Neural nets consist of simple processing elements called "neurons" or "units" which respond, according to an "activation function" to weighted sums of input patterns or vectors **x**. We call the collection of all the weights the weight vector **w**. The outputs of the neurons become inputs of other neurons until a response eventually emerges on a set of designated output units. The network response is thus a function of the network inputs and the weight vector $o_k = F(\mathbf{x},\mathbf{w})$. Given a collection of P input patterns and associated target outputs t_k on k = 1,....K output elements, an error function

$$E_{error} = \sum_{p}^{P}\sum_{k}^{K}\{t_{kp} - o_{kp}\}^2$$

can be computed. The task is to minimize E_{error} by optimizing over the weight vector **w**.

When there are no constraints on the architecture of the neural net, this optimization task can be enormously time consuming. We therefore consider in the following "feed-forward nets" with N input units, one or several layers of hidden units, and one or several output units. A "N - M - L - K" net, for example, is a net with N inputs, M hidden units in the first layer, L units in the second, and K output units.

The input z to each unit in the network is a linear weighted sum of its inputs. Let o_{ij} denote the output of the i^{th} unit in the j^{th} layer and w_{ijk} denote the synaptic weight of the k^{th} unit in the jth layer with input $o_{i(j-1)}$ from the i^{th} unit in the $(j-1)^{st}$ layer. Then

$$z_{jk} = \sum w_{ijk} o_{i(j-1)}$$

is the weighted sum of inputs to the k^{th} unit in the j^{th} layer. The thresholds are represented by w_{0jk} and by convention $o_{0(j-1)} = -1$.

The output of each unit is determined by a nonlinear function of its weighted inputs called the *activation function.* In McCulloch and Pitts neurons, the activation function is a Heavyside step function. Neural networks trained with the back-propagation learning rule typically employ a sigmoid activation function, given by the "squashing function":

$$f(z) = \frac{1}{(1 + exp(-z))} .$$

The sigmoid activation function allows the calculation of partial derivatives necessary for gradient descent optimization.

The dimension of the weight space is the sum of all the weights and thresholds in the net. Training consists of a series of weight changes $\Delta \mathbf{w}$ until the squared error function E_{error} is reduced to an acceptably small value. The algorithm has to find a path in weight space, descending from and weaving through, graphically speaking, a very ragged mountain landscape (defined by the objective function $E_{error}(\mathbf{w})$). It has to overcome not only local minima but find its way out of saddle points in which escape directions are extremely rare. (This problem can become exponentially worse with the dimension N and thus is poorly represented by 3-dimensional intuition).

The back-propagation algorithm computes:

$$\Delta w(n) = \lambda \nabla E + \mu \Delta w(n-1)$$

where ∇E is the gradient of the error function, λ is the learning rate (step size parameter) and $\mu \; \Delta w(n-1)$ is called a "momentum term" that adds a portion of the previous increment $\Delta w(n-1)$ to the gradient. The momentum term is thought to help to overcome local minima. For some problems, conjugate gradient methods have proven faster than backpropagation (Johansson, Dowla and Goodman 1990).

Gradient descent is the method of choice in countless applications. Thanks to back-propagation, neural nets have made a comeback, Hinton, Sejnowski and coworkers succeeded where Rosenblatt (1962) did not. However, gradient descent and its variations (such as conjugate gradient techniques) have intrinsic mathematical limitations (besides being computation intensive and neurobiologically implausible).

Gradient methods are based on linearization. The gradient is the normal to the tangent plane to the level surface (hypersurface) of the objective function. The equation of the tangent plane is the linearization of the objective function. When the nonlinearities dominate then the linear approximation of the objective function is of little use in predicting its values elsewhere. In many important applications, especially when the dimension of the parameter space is high, gradient methods have been a failure.

For difficult functions a variety of optimization methods have been suggested, among others: Genetic algorithms (Holland 1975; Bremermann 1968; Ackley 1987) stochastic optimization algorithms (for a recent survey see Cutler 1988) and a partially stochastic algorithm that interpolates the objective function by means of polynomials on random rays (Bremermann 1970).

2.2.2 SEARCH ALGORITHMS

Every optimization algorithm combines prediction with search. During a search one evaluates the objective function at chosen points. The points can be chosen systematically or at random. When the objective function is easy to predict (through interpolation or extrapolation), then search is less important. Back-propagtion is based upon the first order Taylor approximation of the objective function, which is only locally predictive. We tried to predict the objective function more globally, by polynomial and spline approximation along randomly chosen lines. This method can be an effective method of optimization (Bremermann 1970, Cutler 1988), but in our experiments with training nets to classify n-bit binary strings for "parity" the "landscape" defined by the objective function is so rugged that the approximations in question were useless.

When prediction is impossible, then search is all that remains. In a fully connected neural net we are talking about very large dimensions of the weight space, even for nets with few units. Our algorithms thus must search spaces of high dimension and exhaustive search is out of the question. This leaves only one option: Search through random sampling.

In our case, we have for any given point w^0 a set of "target" points. These are the points to which the algorithm can jump. They are the points $w^0 + \Delta w$ such that $E(w^0 + \Delta w)$ is less than $E(w^0)$. If the objective function is effectively unpredictable, then the sets of target points, averaged over the weight space, have an average density which is Gaussian. The proof is analogous to the proof of the central limit theorem.

Given a random density of targets, how should the search effort be allocated? There exists an elaborate theory for the allocation of search efforts for targets whose location is given by a probability distribution (Stone 1975). In our case it implies that the search effort should be proportional to the target density. In other words, the trial steps Δw should follow the same Gaussian that represents the average target density. The variances of coordinate components of the Gaussian a priori need not be the same, but they can be normalized by scaling.

This leaves a single parameter to be determined: the common standard deviation. We generated random steps from a Gaussian distribution with standard deviation equal to one, then multiplied them with a stepsize parameter h. This parameter can then be adjusted interactively or automatically by means of feedback from successful steps (see section 2.2.5).

2.2.3 MUTATIONAL VERSUS GAUSSIAN RANDOM WALKS

Recently a number of authors have studied "mutational random walks in rugged landscapes", among them Kauffman and Levin (1987), and in the context of macromolecular evolution and the origin of life: Eigen (1988); Eigen, McCasgill and Schuster (1991); Schuster and Swetina (1988); Fontana, Schnabl and Schuster (1989); and Schuster (1985). Macken and Perelson (1989), Perelson and Kauffman (1991) and Macken, Hagan, and Perelson (1991) have explored random walks on "rugged landscapes" that are generated by single point mutations in the context of affinity maturation of antibodies. The mutations affect the coding region for the variable part of antibody molecules, the quantity selected is the binding affinity of antibodies for the antigen to which they respond. During the primary immune response those clones of the B-cell repertoire are expanded whose immunoglobulin receptors have the highest affinity for the invading antibody. Subsequently the affinity (equilibrium binding constant) of antibodies for the immunizing antigens increases, typically ten- to fifty-fold. This is due to *somatic point hypermutations* which occur at the high rate of 10^{-3} per basepair per generation. After an initial improvement of affinity and typically 6-to-8 mutations in the V- (for "variable") region of the immunoglobulins further mutations tend to lead to no further improvement in affinity.

Macken and Perelson have simulated this process by representing genomes by *binary strings* and assigning affinities at random. This gives rise to a "worst case scenario", a totally *rugged landscape*. They consider only single point mutations and describe the random walk as follows:

> "The germline sequence defines a starting point on the landscape. At any stage on a walk, single-mutant variants of the current antibody are tested in random order until the first neighbor having a higher fitness is attained. The walk then moves to this new point in sequence space (i.e. new antibody sequence), and the testing process starts anew..... If no fitter variant is found among these different one-mutant neighbors, the process stops, as the walk has reached a local maximum."

They then analyzed the probability that a random walk ends (gets trapped) at the kth step. This depends, of course, upon the number of mutable bases involved. Macken and Perelson estimate that for the V-region about 700 basepairs are involved, and that about 75% of the mutations (due to the degeneracy of the genetic code) lead to changes in affinity, which amounts to 1575 one-mutant neighbors. Rounding this figure to 1500 they found that the "trapping probability" reaches a maximum at seven steps, and that very few of these mutational walks extend beyond fifteen steps.

When the underlying space consists of digital strings (as it necessarily does in protein evolution), then there is little that can be done about trapping. However, when the underlying space consists of vectors of real variables, then it is possible to draw random steps to from a *spherically symmetric* probability distribution. The n-variate Gaussian (with equal variances) is spherically symmetric, and it has the added advantage to vary *step length* in a way that is an optimal search strategy when the "points of opportunity" themselves have a Gaussian distribution (previous section).

2.2.4 IMPORTANCE OF THE SEARCH-PROBABILITY DISTRIBUTION.

When extending mutational random walks to a vector of continuous real variables, we are presented a choice as to the form the mutation, or search vector, should take. Perhaps the simplest choice would be to choose a random vector from a uniform probability distribution. However, genetic algorithms using a uniform distribution generate highly non-symmetric distributions. Mutation and recombination in a *population* of points is another matter, but in our context the cost of function evaluation makes working with populations unfeasible.

The importance of the Gaussian distribution, especially in high dimension, can easily be demonstrated by substituting another distribution such as a uniform distribution. If we choose the individual trial modifications of the synaptic weights independently from a *uniform distribution* we obtain in N-space a distribution that is uniform over the N-cube. These directions are *not spherically symmetric*. The diagonals of the N-cube are longer than the axes. Deviations from the Gaussian distribution have no serious effect when the dimension is small. They become more and more serious as the dimension increases. The N-sphere inscribed in the N-cube contains the subset of spherically symmetric directions. Its volume V is

$$V_{\text{N-sphere}} = \pi^{N/2} r^N / \Gamma((N+2)/2)$$

Here Γ denotes the Gamma function. Note that $\Gamma((N+2)/2) = (N/2)\,!$ when N is even. Let $N/2 = M$. The volume of the N-cube is $2^N = 4^M$. The ratio of the volumes goes to zero as $(\pi/4)^M/(M!)$. Hence random directions taken from the N-cube are biased away from

	3-Dimensions	N-Dimensions
Cube	$(2r)^3$	$2^N r^N$
Sphere	$\frac{4\pi}{3} r^3$	$\frac{(\sqrt{\pi})^N r^N}{\Gamma((N+2)/2)}$

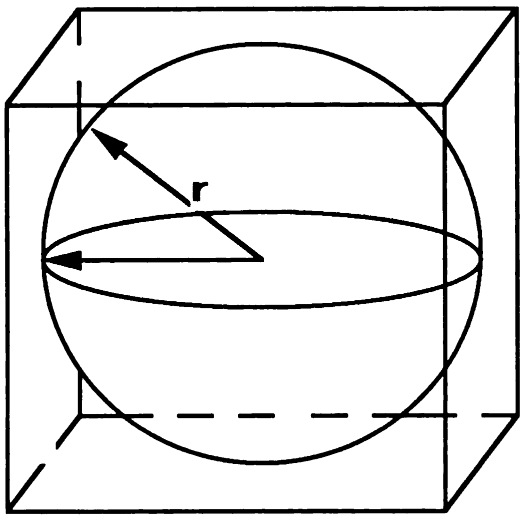


Figure 2.1: Uniform versus Gaussian distribution of random directions.

Vectors in multidimensional space whose components are drawn from a uniform distribution will fill an N-cube. The 3-dimensional case is illustrated. For higher dimensions, the ratio of the volume of the N-cube to the N-sphere becomes astronomical. Generation of a spherically uniform distribution by discarding vectors from an N-cube that lie outside the N-sphere therefore becomes impossible. Components drawn from a Gaussian distribution, however, result in a spherically symmetric distribution of vectors.

spherical symmetry, with the unbiased directions loosing out to biased directions exponentially with increasing N (Figure 2.1).

Robert Hecht-Nielsen has also compared random variables drawn from cubes and spheres in high dimensional spaces. He notes that the length of the diagonal of an N-dimensional hypercube is $(\sqrt{2})^N$ whereas the radius of a sphere remains constant. He concludes that "... cubes in high dimensions are best thought of as spherical porcupines" (Hecht-Nielsen 1990, pp. 42-43).

2.2.5 LEARNING THE STEPSIZE

Besides biased random walks as described above we have also experimented with an algorithm that resembles bacterial chemotaxis. It is a shade less pessimistic about the predictability of the objective function. As before, random trial steps are taken with a Gaussian distribution, but when the algorithm takes a successful step, it attempts to keep going in the same direction until the objective function no longer decreases. In other words this algorithm will attempt to follow a declining slope if it finds one.

The steps of our algorithm are analogous to the motions that bacteria perform during chemotaxis (Koshland 1980, Berg 1983). Bacteria are too small to be able to measure spatial concentration gradients of chemo-attractants. When swimming in a medium with varying concentrations they generate random directions instead and keep going as long as concentration increases. If attractant concentration does not or no longer increases, then they stop, tumble, then emerge in a new direction at random angles to the old direction. In this way they move towards larger and larger concentration values of the attractant. In other words, they optimize the function that describes the concentration of the chemo-attractant in the medium.

How much distance do bacteria have to swim under chemotaxis versus gradient ascent? That depends upon the shape of the equiconcentration surfaces in the medium. When these are spherical (a condition that is ideal for gradient descent), they have to swim an average of only 39% longer as compared with following the gradient (Bremermann 1974). In our case the error function takes the place of the chemo-attractant concentration, and the weight space (of high dimension) corresponds to the 3-dimensional physical space in which bacteria swim.

Algorithms, involving random walks, have long been the numerical method of choice for many particle problems of mathematical physics. "Monte Carlo" algorithms were introduced by Metropolis and Ulam in 1949 (Metropolis and Ulam 1949, Shreider 1966). Recently Ceperly and Alder (1986) reported that the many particle Schroedinger equation can be solved by a "quantum Monte Carlo method" that solves the equation by random walks in the many-dimensional state space. The method is faster and more accurate than conventional variational methods, which for high dimension are too slow and cannot be refined to give sufficiently accurate results. In the limit, for short stepsize and many trajectories, random walks are described by the *diffusion equation* (Comp. Koshland 1980; Berg 1975, 1983; Okubo 1980). Chemotaxis, too, in the limit is described by a reaction-diffusion equation (Keller and Segel 1970; Alt 1980; Nossal 1980). Equations for selection-diffusion have been analyzed by Ebeling et al. (1984). Ceperly and Alder note that the many-particle Schroedinger equation, written in Euclidean time, is equivalent to a bacterial chemotaxis equation.

2.2.6 THE CHEMOTAXIS ALGORITHM

In the Gaussian random walk algorithm, the network is initialized with an an arbitrary set of weights, $\mathbf{w}^0$, and performance $E(\mathbf{w}^0)$ is evaluated. A random vector $\Delta\mathbf{w}$ is chosen from a multivariate Gaussian distribution with a zero mean and a unit standard deviation. This random vector is added to the current weights to create a 'tentative' set of weights w^t :

$$w^t = w^0 + h\Delta w$$

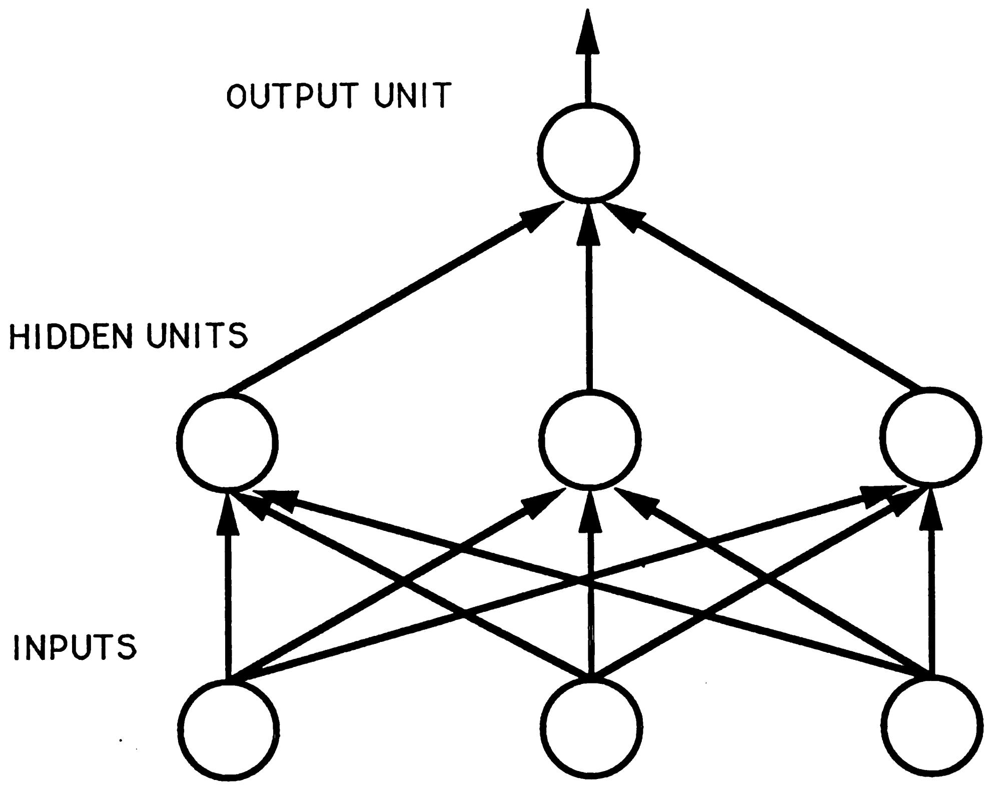
where h is a stepsize parameter. Performance $E(w^t)$ is then calculated for the tentative weights. If the error of the new configuration is lower than the original configuration, the tentative changes in the weight vector are retained; otherwise, the system reverts to its original configuration. A new random vector is chosen, and the process is repeated.

In the chemotaxis algorithm, if a successful direction in weight space is found, an additional step is taken along *the same random vector* and the stepsize parameter, h, is increased automatically. Weight modifications continue along this same random vector until progress ceases. In other words, the algorithm will attempt to follow a declining slope if it finds one. In addition, if after several trials, a successful direction is *not* found, h is reduced by 50%. This allows for automatic adjustment of the single learning parameter, which otherwise has to be found empirically. Because of these automatic features and the ease of implementation, the chemotaxis algorithm was used in all computer simulations reported here.

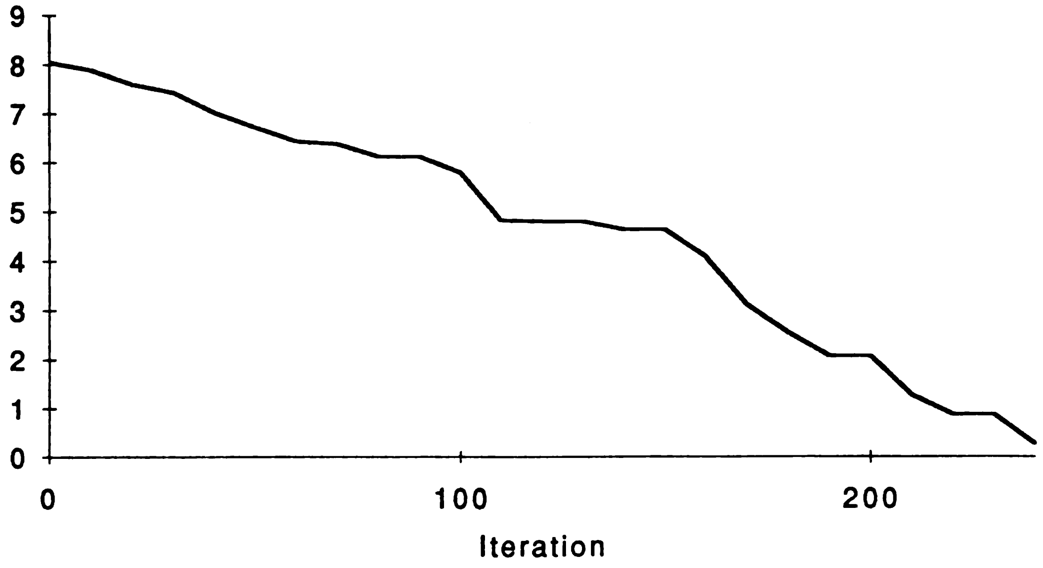
Figure 2.2 (Following Pages): Graphical presentation of the random walk in weight space.

A 3-3-1 network was trained using the 'chemotaxis' algorithm for 3-bit parity. Training continued until the network responses to all 8 patterns were within10% of their 'target' values. Total system error versus iterations of the algorithm is shown in figure 1B. Figure C shows the evolution of the individual weights during training from the three hidden units and bias to the output unit. Each marker identifies a point in training where a successful random step was found. Evolution of the weights from the input nodes to the three hidden units are shown in figures D-F. Random walks in the weight space of larger nets look similar, but would require too much space for graphical presentation.

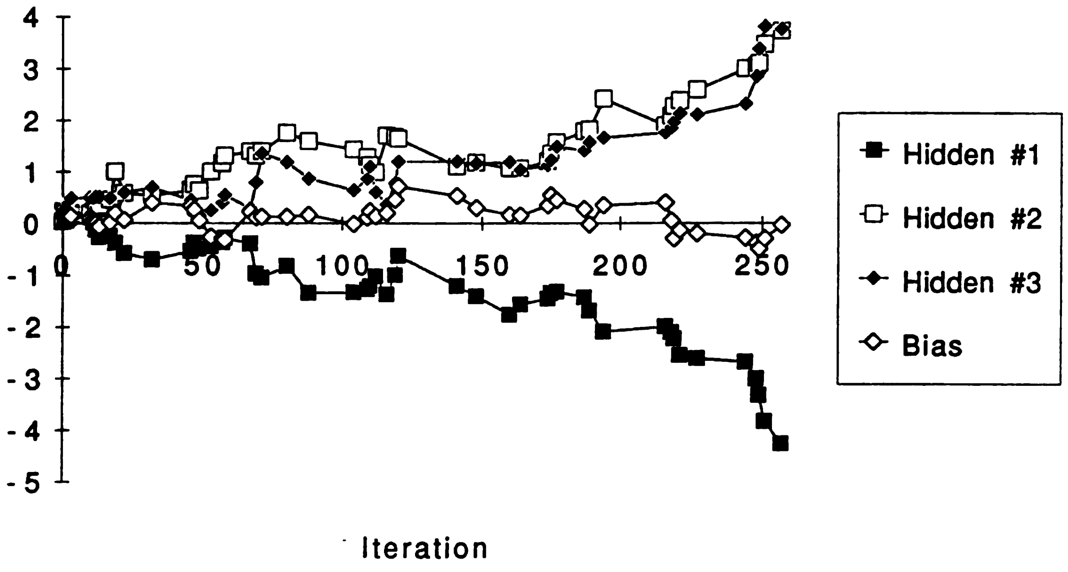
(A)



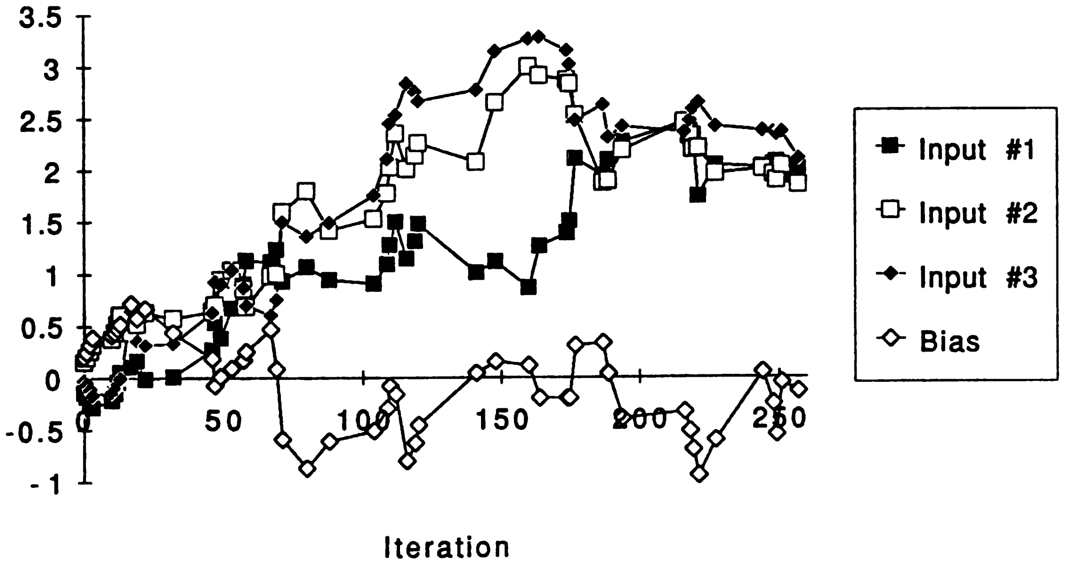
B: TOTAL SQUARED ERROR



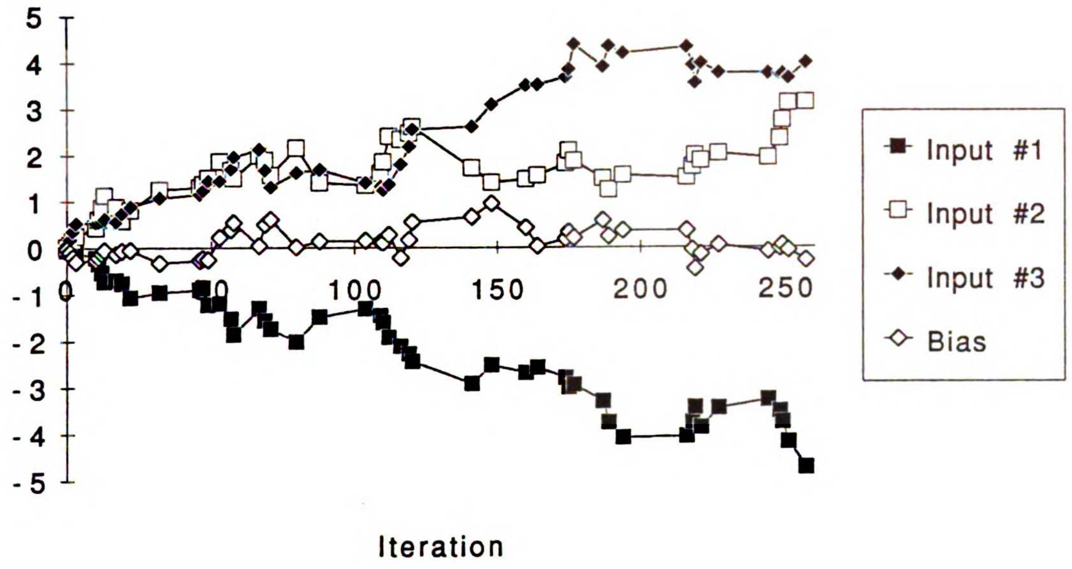
C: WEIGHTS INTO OUTPUT UNIT



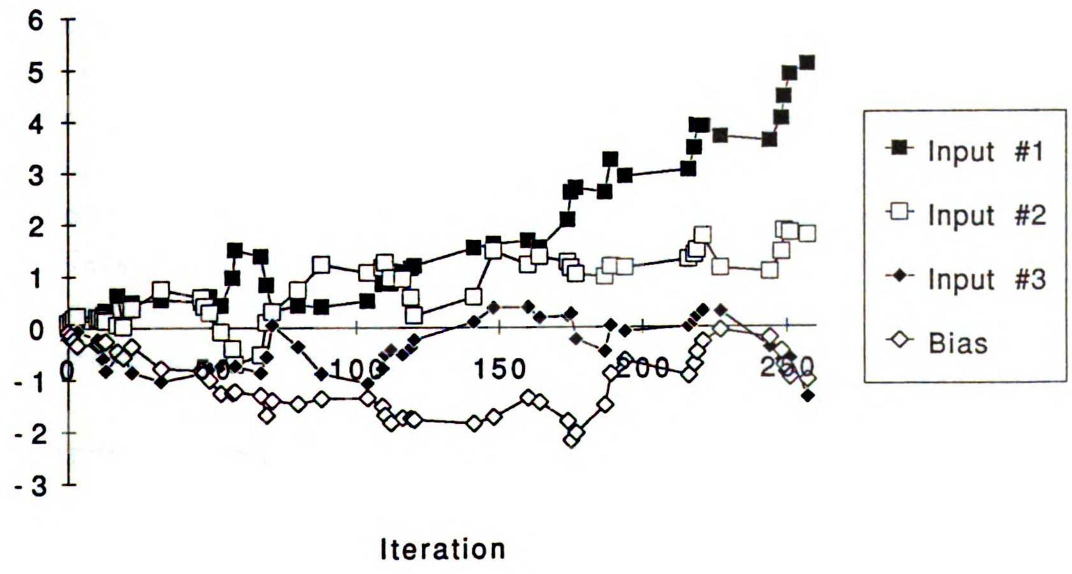
D: WEIGHTS INTO HIDDEN UNIT #1



E: WEIGHTS INTO HIDDEN UNIT #2



F: WEIGHTS INTO HIDDEN UNIT #3



2.2.7 ACCELERATED GUIDED RANDOM SEARCH

It should be noted that optimization through random walks is not new. Two years after we developed the chemotaxis algorithm, Daniel Chester (U. of Delaware) pointed out to us that the chemotaxis algorithm is similar in spirit to a method which was known as the *accelerated guided random search*: (GARS) developed by Lewey O. Gilstrap, Jr. and others at Adaptronics, Inc. (McLean, VA):

> " [T]he accelerated random search begins by exploring the vicinity of its initial estimate. The random trials are governed by a normal distribution of probabilities which is centered on the initial point. ... the accelerated random search follows an unsuccessful random step, with a step of equal magnitude in the opposite direction. By this means, a successful step is usually achieved on the second trial if not on the first random trial. ... A successful step is always followed by another step in the same direction ... each successive step is given double the magnitude of the prior step."(Barron 1968)

Barron (1968; 1970) used GARS to optimize control parameters in flight control systems. Mucciardi (1972) even applied GARS to neural net-like classification structures called "neuromine nets". Mucciardi's paper presented an analysis of neuromine nets and the algorithm, but provided only simple examples of its application. Advanced applications of the GARS algorithm never materialized due to the lack of available computer resources. Their failure to apply the algorithm to the complex classification problems emphasized in *Perceptrons* kept GARS from receiving much attention.

Another aspect of random search, overlooked in by the group at Adaptronics, was it's biological plausibility, discussed in the next section.

2.3 RANDOM WALK LEARNING IN BIOLOGICAL NEURAL SYSTEMS

2.3.1 GAUSSIAN FLUCTUATIONS OF NEURONAL CONNECTIONS AND SYNAPTIC STRENGTH

A Gaussian distribution can easily be generated in biological nerve nets for two reasons:

1) The product of individual Gaussians (with standard deviations σ_1 to σ_n) is the multivariate Gaussian distribution with multivariate standard deviations σ_1 to σ_n . In other words if individual synaptic weights fluctuate *independently*, each according to a Gaussian distribution, then the entire weight vector fluctuates according to a multivariate Gaussian. Not only is the multivariate Gaussian distribution required by the the optimal search theorem, but it can be generated *without interaction between the weight changes.*

Glanzman et al. (1990) have observed structural change *in vitro* of a coculture of *Aplysia* sensory neurons and their target (L7 motor) neurons over a 24 hour period. They note that morphological changes (varicosities and new synaptic processes) at the junctions between the sensory and motor cells appeared to be normally (Gaussian) distributed in the control group. In the cocultures repeatedly treated with serotonin, however, structural change was shown to be highly biased toward increases in connectivity. Furthermore, they showed that these observed structural changes corresponded to measurable changes in monosynaptic EPSP (excitatory postsynaptic potential) produced in L7 motor cells by firing the sensory neuron. Perhaps the random variation observed in the control group serves a vital role in learning by generating tentative, exploratory variation in neural structure. (see section 2.7)

2) When synaptic strength is the result of many components (channels in the presynaptic membrane, transmitter release mechanisms, postsynaptic receptors, etc. [Lynch 1986; Kennedy

1988]) then fluctuations in the total mechanism, because of the central limit theorem are distributed approximately Gaussian, even if individual fluctuations are not.

Learning is likely to occur not only at the level of individual neurons but could involve "groups" or populations of neurons (Edelman 1987). Plasticity in contact between axons and spines (for which there is evidence, e.g. [Aoki et al. 1988]) would lead to fluctuations in the strength of contact between groups. Such fluctuation, because of the central limit theorem, would also follow a Gaussian distribution. *Therefore the same basic learning algorithm would apply to learning by individual units as well as learning by large and small groups of neurons.*

2.3.2 FIXATION OF SUCCESSFUL MODIFICATIONS

Our algorithm requires that all fluctuations in the strengths of weights (or synaptic contact between groups) remain tentative and reversible unless made permanent by a special "hold" signal (which is analogous to Grossberg's postulated "print" signal [1988]) It is important to note that this signal does not require any local computations whatsoever; it is a global signal addressed "to all synapses" within the net. It could be an electrical signal propagating through the net (such as tetanization), or chemical, or an interaction of both. This signal would fit Crick's (1989) description of a signal "to tell the system when something is worth remembering". He points out that "there are many return pathways in the brain." The global signal is consistent with his observation of the existence of "*diffuse pathways*, such as that from the locus coeruleus, ... [in which] ... one such neuron sends much the *same signal* to many parts of the brain."

Our proposed modification rule also seems to be consistent with current knowledge about molecular mechanism of synaptic modification (Kauer et al. 1988; Stevens 1989; J.H. Williams, et al. 1989). Both back-propagation and our algorithms require that synaptic strength can be increased as well as decreased. Long-term synaptic change in *Aplysia* sensorimotor synapses can be produced by a cellular analog of classical conditioning (Buonomano and Byrne 1990). Associative long term depression (LTD) has been reported in the same system by Stanton and Sejnowski (1989). They show that this is consistent with a "Hebbian covariance learning rule", but such phenomenon are consistent with several reinforcement learning rules - including random walks.

2.3.3 ORIGIN OF THE GLOBAL SIGNAL

The algorithm requires that the performance of a net be evaluated. This evaluation could be accomplished by other brain circuits. We do not consider this requirement problematic, since evaluation of performance tends to be computationally easier than improvement. For example, throwing a ball requires precise coordination and timing of numerous muscles. Good performance is hard to achieve and may require extensive training. But, how close a ball comes to hitting the target is relatively easy to determine. Evaluation of accuracy can be processed separately by the visual cortex - independent of networks involved in generating the movement. One portion of the brain thus could act for another system as "supervisor".

Other learning schemes also require a global signal. In the Boltzmann machine weights are adjusted according to the activities of the neurons that are connected (Ackley et al. 1985). The local adjustment mechanism, however, must receive information when the neural net is free running and when it is "clamped". Barto et al. employ a rather complex 'adaptive critic element' in their reinforcement learning schemes (Barto, Sutton and Brouwer 1981; Barto, Sutton and Anderson 1983). In Edelman's "Neural Darwinism" (1987) a global "value" of the net activity is required.

2.4 SIMULATION EXAMPLES: TRAINING NETWORKS ON BENCHMARK PROBLEMS[2]

2.4.1 N-BIT PARITY

Here we report results of computer simulations of the chemotaxis algorithm applied to the on the "N-bit parity" problem. The parity value of N binary inputs is defined as 'true" for an odd number of "true" inputs and "false" otherwise. N-bit parity can be considered a generalization of the "exclusive or" (XOR) which is known to be problematic for single layer perceptrons (Minsky and Papert 1969). Parity is considered a "difficult" test case for neural networks since its solution requires complete knowledge of the input vector in order to determine a correct response. In addition, neural network training time is known to scale exponentially with N (Tesauro 1988). However local minima do not seem to present a significant obstacle to training when more than the minimal number of hidden units are allowed. Michael Conrad has also noted evolutionary optimization is easier on a system with redundant variables (Conrad 1983).

The network used for this study had an 'N - (2N+1) - 1' architecture where N represents the number of bits in the input layer. In all the explorations reported in the following binary inputs were represented as 'one' and 'minus one'. In this way we avoid multiplying weights by zero.

The output unit's response to a given input is defined as 'perfect' if it is within 0.2 of the the target output. A training cycle consisted of testing all 2^N input vectors on a given trial weight vector. Training was continued until 100% of the input vectors had

[2]An independent comparison of the chemotaxis, back-propagation, recursive error minimization, second-order back-propagation and cascade correlation training algorithms on benchmark problems appears in Wilson (1991).

'perfect' responses. Best, worst and average training times as a function of input vector dimension are shown in Table 2.1.

2.4.2 DISCUSSION

These training times are roughly an order of magnitude longer than back-propagation. Note that the adjustment of weights according to back-propagation is computation intensive, while ours is trivial. A back-propagation cycle requires about twice the amount of computation as our algorithm. Also, Tesauro and Janssens fine-tuned their version of back-propagation by choosing optimal values for the learning rate and momentum parameters for each N. This required considerable trial and error. All together they expended several weeks of SUN workstation CPU-time compiling their statistics (Table 2.1). There are indications that training times for our method could be further improved through fine-tuning of the algorithm. For example, one could add a "momentum term" $\mu \Delta w(n-1)$, where $\Delta w(n-1)$ is the previous weight adjustment. Chemotaxis adjusts the stepsize automatically. Automatic expansion and contraction of net architecture might also help improve convergence (see below).

We also have experimented with back-propagation by substituting it for our own optimization algorithm, minimizing the *sum* of squared errors of *all* test patterns rather than, as Tesauro and Janssens, adjusting weights after each *individual* test pattern presentation. Tesauro and Janssens' performance figures are much better than the results that we obtained in this way with constant learning rate λ and momentum parameter μ). Tesauro and Janssens' method depends critically upon λ and μ . We did not attempt to fine-tune these parameters, which would have been very computation intensive. Our "chemotaxis algorithm", in contrast, is automatic and does not require any tuning.

Back-propagation likewise can be modified to automatically adjust parameters. When we incorporated a rule for adjusting the learning rate, (omitting the momentum term), back-propagation

Table 2.1: Comparison of the Chemotaxis Algorithm and Back-Propagation*

Dimension (N)	Chemotaxis Average (Low/High)	Back-Prop (cycles through training set)
2	113 (35/188)	24
3	251 (96/421)	33
4	962 (649/1660)	75
5	1259 (820/1951)	130
6	4169 (2494/7107)	310
7	5789 (3159/10169)	800

* Average, best and worst training times for the chemotaxis algorithm. Net architecture: N-(2N+1)-1. Weights were updated after performance was evaluated over the entire input set.

Back-propagation averages from Tesauro and Janssens (1988). For comparison of computational effort, the figures of Tesauro and Janssens should be doubled (see text).

performed slightly better than our chemotaxis algorithm on 4-bit parity.

Finally, we note that many researchers introduce the equivalent of a stochastic element into the back-propagation algorithm. They do not minimize (as we do) the objective function that is generated by summing the squared output errors over the entire set of inputs. Instead they compute the gradient and adjust the weights after each individual input sample (e.g. Tesauro and Janssens 1988). Therefore the objective function changes after each sample presentation. In this case, the "momentum term" actually is serving as a "memory" term - to remember the landscape of previous samples. Since for the parity problem the objective function is rather erratic, the resulting gradient directions would seem to behave in a pseudo-stochastic way.

2.4.3 THE 4-BIT ENCODER PROBLEM

This problem was introduced by Ackley, Hinton and Sejnowski to demonstrate a learning algorithm for Boltzmann Machines (Ackley et al. 1985). It has since become a standard problem in neural networks (see also Rumelhart, Hinton and Williams 1986; Ballard 1987, Hinton 1988, Crick 1989). The net architecture chosen for this study takes the form 4-N-4: four input nodes are connected to N hidden units, which then feed into four output units (In this study N=2,3,4). Four input patterns: (0,0,0,1), (0,0,1,0), (0,1,0,0), (1,0,0,0) are then trained to elicit a corresponding set of patterns at the output units. For N less than 4, the network is forced to find a more compact representation of the input vector in order to communicate all of the input patterns to the output layer, thus the name *encoder*.

Twenty training runs were performed for each architecture (4-4-4, 4-3-4, 4-2-4). Weights were updated after each cycle through all four input patterns. Training was continued until the sum of the squared errors had been reduced to 0.1. All runs converged except

Table 2.2: Performance of the Chemotaxis Algorithm on the Encoder Problem with Communication "Bottlenecks" (1)

Architecture	Average Training Times	
	(to 1.0)	(to 0.1)
'4-4-4'	259	496
'4-3-4'	352	648
'4-2-4'	328*	963*

(1)Average performance on the encoder problem as described by Ackley et al. (1985). In this problem, the hidden units are fewer in number than the number of input and output units (forcing coding in the hidden layer). Averages are taken from 20 simulation runs. One run of the 4-2-4 encoder failed to learn all 4 patterns and was omitted from that average.

one run of a 4-2-4 net, which learned only 3 of the four patterns. Average training times are given in Table 2.2. Restriction of the number of hidden units did not appear to present a significant obstacle to the chemotaxis algorithm in this case.

2.4.4 THE MULTILAYER ENCODER

The problem was extended to *several* layers of hidden units (Ballard 1987). Following Ballard's convention, the architecture chosen consisted of four input units, feeding into four hidden units, and into N-1 layers of hidden units. Note that no 'bottleneck' was imposed for simplicity. Performance was averaged over 20 training runs for networks with 1 to 5 layers of hidden units (3 to 7 layers of nodes).

The algorithm converged for all runs. Average training times to a squared error of 1.00 are given in Table 2.3 (for 1 and 3 layers of hidden units, errors were further reduced for comparison). Comparison with Ballard's study shows that the chemotaxis algorithm is 2 to 5 times slower than back-propagation for *one* hidden layer. But, *for nets with three or more layers of hidden units, our method matches or outperforms back-propagation.* A comparison with an evolutionary optimization algorithm on this problem is given in (Smalz and Conrad 1990).

2.5 VARIATIONS OF THE BASIC ALGORITHM

2.5.1 BOLTZMANN FACTOR

In the spirit of the Metropolis (Metropolis et al. 1953) algorithm and the Boltzmann machine (Hopfield 1984; Hopfield and Tank 1985, 1986; Ackley, Hinton, and Sejnowski 1985; Geman and Geman 1984/88; Boseniuk, Ebeling and Engel 1987; Ackley 1987; Haken 1988; Day and Camporese 1990) we modified the algorithm such as to allow the occasional acceptance of values of the objective

Table 2.3: Comparison of Training Times for the Multilayer Encoder Problem*

#of layers of Hidden Units	Chemotaxis Algorithm Training (to 1.0)	(to 0.1)	Back-propagation Training times to squared errors: (to 1.0)	(to 0.1)
1	259	479	50	200
2	584	-	-	-
3	829	1837	2000	3000
4	2404	-	-	-
5	3560	-	-	-

*Averages taken over 20 simulation runs.
Back-propagation numbers are estimated from Ballard (1987).

function that were *higher* than the current value in the hope that this helps overcome local minima. Acceptance of higher values was governed by a probability equal to a negative exponential of a "temperature" factor and the difference in values of the objective function (Boltzmann factor). Unfortunately this did not speed up convergence but had detrimental effects, allowing a slow deterioration of the function value, unless the "temperature" factor was virtually zero, which takes us back to the original algorithm.

2.5.2 ACTIVATION FUNCTIONS

Rosenblatt's (1962) original perceptron model is very similar to the feed-forward nets that we have studied, except for the sigmoid activation function, (sometimes called "squashing function"). He used a step function instead of a sigmoid, in accordance with the McCulloch and Pitts neuron model. The sigmoid function was originally introduced in Boltzmann machines, where it represents firing frequencies of neurons (Hopfield 1984; Ackley, Hinton and Sejnowski 1985). Subsequently Rumelhart, Hinton and Williams (1986) used it as their *activation function* with back-propagation in feed-forward nets. Sejnowski and coworkers successfully applied this methodology to several non-trivial problems (see Introduction).

The sigmoid is convenient for applying back-propagation, which computes the gradient of the performance (error) function. The gradient contains the derivative of the activation function. In the limiting case the sigmoid function becomes the discontinuous step function $H(t)$, which is zero for $t < 0$, $H(t) = 1$ for $t > 0$. The derivative of $H(t)$ is the Dirac delta function: $H'(t) = \delta(t)$. The delta function is zero for $t \neq 0$, but $\delta(0) = \infty$, and thus the formula for back-propagation does not work.

The sigmoid function may be difficult to justify as the response function of an individual neuron. However, if the "units" are not individual neurons but populations of neurons then the collective output could easily be a sigmoid or some sort of a saturation curve resembling it. It may thus be of interest to know that the exact shape of the group response curve does not seem to matter.

The exact shape of the activation function is not critical for our algorithms. In some experiments we have replaced it with a piecewise linear "ramp" function without affecting performance of the algorithm. (On the contrary, ramp functions are cheaper to compute than sigmoids, which involve exponentials). McClelland and Rumelhart (1981) and Rumelhart and McClelland (1982) have employed this activation function in connectionist models.

Rumelhart, Hinton, and McClelland (1986) compare different types of activation functions that have been proposed for connectionist models as well as feed-forward nets. Among these are *conjunctive units* of Feldman and Ballard (1982) that have terms that are the *product of inputs*, the same as the "sigma-pi" units of Williams (1986). We have experimented with such units in the N-bit parity problem and found that our algorithm converges faster when products of nearest neighbors are added as inputs to the weighted sums. This shows that our algorithm can handle activation functions of many kinds without any added computational complexity. In contrast, *computation of the derivatives of activation functions other than sigmoids* (which is required for back-propagation), can become very complex. (Note that the sigmoid function is special because $f' = f(1 - f)$). We have not collected performance statistics for Feldman-Ballard and Williams-type activation functions since our objective was to compare our algorithm with published performance figures for back-propagation.

2.5.3 NET EXPANSION AND CONTRACTION

We conducted experiments for n-bit parity (interactively, with simple random walks as well as chemotaxis) where during the training of a net additional units are added or removed. This process does not unduly disturb training progress. In fact, when a task proved too difficult for a net (not enough hidden units), adding units would help achieve ultimate convergence. Also, when training reached a prolonged plateau, the algorithm could be made to converge by training a smaller net (on a simpler version of the problem), then expanding the net and training it on the full problem again.

2.6 A CONJECTURE FOR CORTICAL MAPPINGS

The principles of cortical self-organization have recently been reviewed by Stryker et al. (1988). In this context, topographic maps, especially from the retina to the visual cortex (and other areas of the cortex, as many as 24 in the rhesus monkey), are of special importance. Von der Malsburg and Singer (in Rakic and Singer 1988) have suggested that the establishment of the topological map from the retina to target areas in the cortex is "an important paradigm case".

> "There are many topological fiber projections between thalamic areas and also between and within cortical areas. Retinotopy is one of the few biological model problems experimentally addressed with enough intensity that theoretical issues can be decided".

We (HJB and RWA) propose the following three mechanisms:

> "A) There is a mechanism to guide fibers to tectum.
>
> B) There is a mechanism to position fiber terminals within tectum. This mechanism is responsible for rigid constraints on the mapping.
>
> C) There is a fiber-sorting mechanism which improves the precision of the mapping over that attained by mechanism B alone and which is activity-dependent."

They further state that "the fiber-sorting mechanism C, since it deals with activity-dependent network organization....can be generalized to other interesting cases, in particular to neocortex".

A bundle of fibers ascending from the retina, through the optical nerve, lateral geniculate to the visual cortex resembles the "encoder problem" (Figure 2.3).

There are many reverse pathways in the brain (Van Essen 1985). The corticofugal fibers carry information from the cortex to the dorsal lateral geniculate nucleus. Could it be that these fibers return information to the lateral geniculate where it is compared for quality of matching? A topological map is entirely determined by the maps of points and their neighborhoods such that the mappings in neighborhood intersections are consistent. According to postulates A and B by Von der Malsburg and Singer, morphogenesis locates fibres coarsely in correct neighborhoods. The problem thus is to account for refinement of this map. Suppose a cell in the originating layer creates a ring of inhibition when it fires. Then its projection should be surrounded by a "ring of silence". If the point to point map is not consistent in the projection area, a distorted ring will result. Suppose that such distortion would generate an error signal that is returned to the originating layer. The random walk algorithm could then adjust weights and or connections until the error signal is minimized. Repeated over and over again for all projecting points, the initially coarse map could be improved to the limit of resolution.

Another problem is: how do the reverse fibers "know" exactly where to go? Here is a suggestion for a possible solution of this dilemma: *The projections from the lateral geniculate nucleus to the visual cortex and back to the geniculate are trained as a single encoder*. The input is sent on ascending pathways to the cortex and returned on reverse channels to its origin where the original pattern and its "echo" are compared. This configuration would constitute an encoder with the cortex functioning like a layer of "hidden" units. Initially the two patterns might match only very coarsely. The

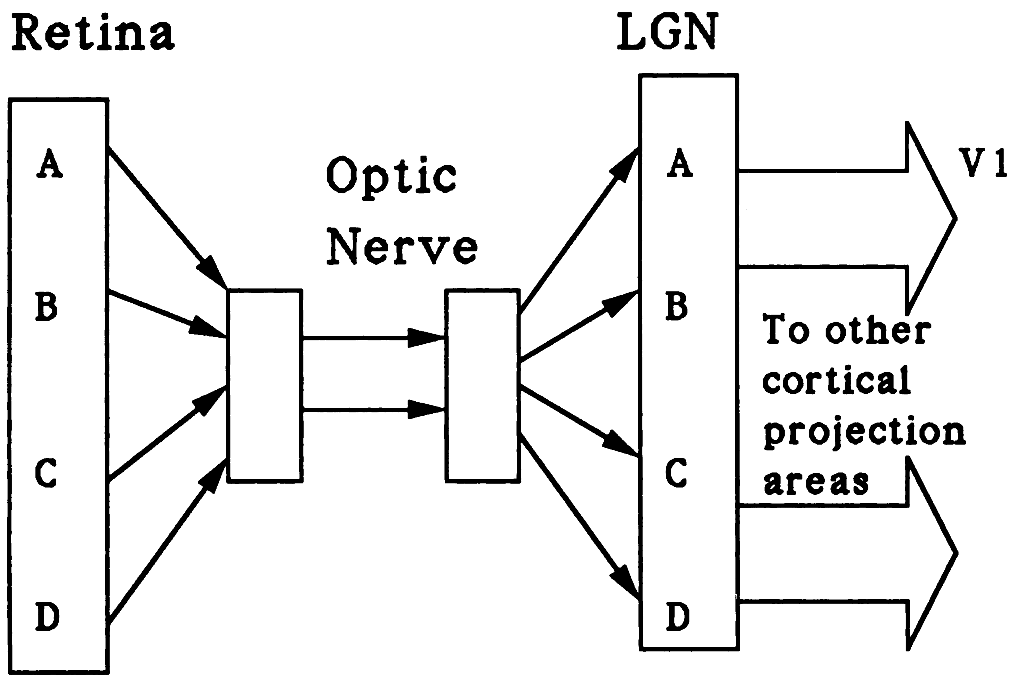


Figure 2.3: Schematic representation of the map from the retina to the lateral geniculate nucleus.

Pathways originating from the retina project via the optic nerve through the Lateral Geniculate Nucleus (LGN) on to the primary visual cortex (V1) as well as other cortical projection areas. The mapping on V1 is a point -to-point map of 2-D visual space. This characteristic organization is typical of most mappings in the brain and resembles a multilayer encoder. (Figure adapted from Roe et al. 1990.)

encoder would then be trained with varying inputs until the identity map becomes better and better.

To ensure that the map is a one-to-one map we suggest that signals originate alternately in the geniculate and being trained to return to the point of origin, and conversely: signals that originate in the projection areas are trained to return to the point of origin in the cortex.

Our model would also be consistent with multiple projections onto cortical areas, each projection being fine-tuned separately. The lateral geniculate would thus be a distribution area for multiple retinotopical maps.

2.7 DISCUSSION: OBSERVABILITY OF BIOLOGICAL LEARNING

What learning mechanisms, if any, accounts for the plasticity in the organization of neural structures? Identification of learning mechanisms is complicated by the problem of *observability*. The only mechanism for systematic synaptic change found thus far is the Hebbian-like changes involved in Long-Term Potentiation and Depression (Chapter 1) which only requires observation of two or three locally interacting neurons. We have argued, however, that Hebbian learning rules are insufficient for learning complex, nonlinear mappings, such as N-bit parity. We have demonstrated that a simple mechanism can be effective in training networks of highly abstracted neurons. Evidence for non-local learning mechanisms such as the Gaussian random walk presented here will be difficult, if not impossible. What we can hope to find is the necessary components of a learning rule. As discussed in Section 2.3, our algorithm requires three components to function:

1. a mechanism for local variation,
2. a structure to evaluate performance, and
3. a reinforcement signal to fix successful modifications.

Recent physiological experiments by Glanzman and others (Glanzman, Kandel and Schacher 1990) can be reconciled with this simple learning rule. Here we briefly describe their results, offer an interpretation consistent with a global reinforcement learning rule, and suggest additional, clarifying experiments.

Glanzman et al. (Glanzman et al. 1990) studied an in vitro coculture of *Aplysia* sensory neurons and their target [L7 motor] cells. The sensorimotor cocultures were grown for 5 days and observed by fluorescence video micrographs. One group of preparations was repeatedly treated with the facilitating transmitter serotonin (5-HT) for 24 hours. Afterwards, the coculture was imaged again to look for structural changes. (Serotonin was not applied to the control coculture.) This study was significant in that they were able to directly image structural changes - rather than relying on comparison of two different populations of neurons. Morphological changes (varicosities or new processes) at the junctions between the sensory and motor cells were rated on a subjective scale. Increases in size or number of varicosities or processes rated increasingly positive scores (up to +2) and decreases rated negative scores (to -2).

In the control group, morphological changes were found to be normally distributed with a mean change of zero on their rating scale. In the cocultures treated with serotonin, however, structural change was shown to be highly biased toward increases in varicosities or processes. Furthermore, they showed that these structural changes corresponded to measurable changes in monosynaptic excitatory post-synaptic potential (EPSP) produced in L7 motor cells by firing the sensory neuron. Thus, they were able to show both physical and electrophysiological facilitation can be induced *in vitro* by a single chemical signal - serotonin. (In a separate experiment, they showed that memory-related structural changes did not occur without the presence of the appropriate postsynaptic target.)

We suggest that the random variations in the control experiment may serve a vital role in learning, that is, to generate new trial connections and efficacies (see section 3.1). The system may make use of these trial-and-error explorations in structural organization as a vital component of the "learning" process. Interpreted in this light, serotonin release in a cluster of neurons may serve as a local "print" (or fixing) signal to retain effective changes. However, the experiment described by Glanzman, et al. cannot differentiate between serotonin's putative role as a generalized *facilitating growth factor* or a *reinforcement* signal.

We propose an experiment in which serotonin is applied *selectively*. With more frequent imaging, structural change could be monitored as a function of time. Images could be taken, for example, every 4 hours. Serotonin would only be applied *if* growth from the previous image had been detected. These applications would serve as a "print", or reinforcement, signal. In the control experiment, the same number of serotonin applications would be applied, but at random intervals. If our hypothesis is correct, we would expect the experimental group to show significantly more structural change. If, however, serotonin is simply acting as a generalized growth factor, there should be no difference between the two groups. Difficulties would arise in interpretation of a negative result as it may be difficult to measure minute changes in structure on a subjective basis. These problems may be overcome by increasing the total duration of the experiment to 3-4 days (instead of 24 hours) and sampling every 8 or 10 hours.

There is other evidence the other transmitters can induce depression of synaptic efficacy (Montarolo, Kandel and Schacher 1988). There is also evidence that the effects of serotonin may be more selective (Eaton and Salt 1989). Its effectiveness is also modulated by secondary messengers (Bergold et al. 1990) and threshold phenomena (Flavahan and Vanhoutte 1988). We note that a more orchestrated and directed learning process would only serve to increase the efficiency of a simple global reinforcement signal. We also emphasize that our proposed learning scheme does not preclude other modes of change (i.e. Hebbian facilitation and conditioned response).

CHAPTER 3

NEURAL NETWORK THEORY

3.1 INTRODUCTION

We have demonstrated that a biologically plausible learning algorithm is capable of training feed-forward neural networks. It is useful to know what these networks can do *in principle* as well as whether a particular training algorithm is effective in exploiting these capabilities. As mentioned in Chapter 1, neural networks are potentially powerful computing devices. Here we discuss the theoretical capabilities of the more limited, feed-forward neural networks. This lays the foundation for a discussion of capabilities of *fully-connected* (recurrent) networks presented in Chapter 5. This mathematical treatment is essential for the understanding of the constraints which govern biological information processing.

Knowledge of the theoretical capabilities and limitations of neural networks is critical for the design of successful applications. As some of these theoretical developments are fairly recent, a general reference on the topics covered here is not yet available. (For an excellent reference on the state of the art up to 4 years ago, see *Brains, Machines and Mathematics* [Arbib 1987].) The topics covered here provide the necessary background for the applications presented in subsequent chapters.

We present a review of three relevant theoretical issues for neural network applications: function approximation, pattern recognition and generalization. We review recent analyses that interpret neural networks as convenient functional approximation devices. As a special case, we discuss the capabilities of neural networks in estimating probability density functions and discriminant functions in problems of pattern recognition. Neural network methods are compared to traditional pattern recognition techniques, especially statistical pattern recognition. We review studies which show that under certain conditions, neural network training algorithms can be used to approximate the Bayes optimal discriminant function. We show that neural networks possess advantages over conventional techniques. Finally, we discuss the limitations of the phenomenon of 'generalization' in neural networks. We argue that the generalization capability of a trained network is critically dependent on the prototypes in the training set and network architecture and that the particular optimization (training) algorithm is mostly irrelevant.

3.2 NEURAL NETWORKS AND FUNCTION APPROXIMATION

In several applications, neural networks have demonstrated their capability as convenient function approximation devices. Recently, theoretical analyses have supported this interpretation of neural network function. The output of each unit in a standard neural network is a nonlinear function of a weighted sum of it's inputs. A two layer, feed-forward network has a mathematical structure where at the output layer, each unit performs a summation of several nonlinear functions of the input vector of the form:

$$\sum_{j=1}^{N} \omega_j \, \sigma\left(w_j^T x + \theta_j\right)$$

were σ represents a nonlinear function, N is the number of hidden units, θ is a threshold, x is the input vector, and w and ω represent weight vectors. Functions of this type have some rather remarkable universal properties relating to the representation of functions and Hilbert's 13th problem.

At the turn of the century David Hilbert, in an address to the Paris Academy, presented a list of over 20 unsolved problems in mathematics. Work on these problems continues to occupy a substantial amount of effort of the mathematical community (Browder 1976). Hilbert's 13th problem conjectured that there are continuous functions of three variables that are not representable by continuous functions of two variables, or simply, that "true functions" of three variables exist (Lorentz 1976). This conjecture turned out to have two answers. "The astonishing result of Kolmogorov" (Lorentz 1976) was that, under certain conditions, Hilbert's conjecture was false. Kolmogorov (Kolmogorov 1957; Lorentz 1976) proved that there exist continuous increasing functions $\Phi_{pq}(x)$, on $I = [0,1]$ so that each continuous function f on I^n can be written in the form

$$f(x_1,\ldots,x_n) = \sum_{q=1}^{2n+1} g_q\left(\sum_{p=1}^{n} \Phi_{pq}(x_p)\right),$$

where g_q are (2N +1) properly chosen continuous functions of one variable. Hecht-Nielsen interpreted Kolmogorov's Theorem in terms of feed-forward networks: "an arbitrary, continuous function from the n-dimensional hypercube to the m-dimensional space of real numbers $\mathbf{R}^m$ can be implemented exactly by a 2-layer network" (Hecht-Nielsen 1987, Hoffgen and Siemon 1990). Hecht-Nielsen was referring to networks where the activation functions of the 2N+1 "hidden units" are chosen to correspond to the g_q's in Kolmogorov's Theorem. But the Kolmogorov representation requires specific and precise nonlinear functions (Cybenko 1989). How close can a neural network composed of generic, nonlinear functions get to this ideal?

In most neural network applications one cannot realistically expect any given training algorithm to find these ideal solutions, so more recent analyses have emphasized the functional *approximation* capabilities of neural networks. Cybenko notes that "requiring a finite linear combination *exactly* represent a general continuous function is typically asking too much [emphasis added]" (Cybenko

1989). In the same paper, however, he proved that linear combinations of sigmoidal functions (which serve as inputs to the output of the second layer of a feed-forward network) can uniformly *approximate* any continuous function. A similar result was obtained by Funahashi (1989) who showed that any continuous mapping can be approximated by a net with one hidden layer, provided that an unlimited number of units are allowed. Networks consisting of non-sigmoidal activation functions (such as Gaussian hidden units) can also be used as universal approximators (Hartman et al. 1990).

Figure 3.1 schematically illustrates a neural network applied to a functional approximation task. One can imagine that the neural network accomplishes the task in much the same way as a function is represented by a Taylor series, Fourier transform or a spline approximation. In fact, Poggio and Girosi offer a complementary interpretation of neural networks, equating neural networks to the technique of generalized splines (Poggio and Girosi 1990).

It is important to note that Cybenko does not guarantee that any particular training algorithm will find a sufficiently close approximation, nor is any indication given as to *how many* hidden units are required to accomplish the desired task, even in principle. Several studies have attempted to establish "least upper bounds" on the number of hidden units needed to realize an arbitrary function (Lippmann 1987, Chester 1990a, Huang and Huang 1991), but nobody has demonstrated a training algorithm capable of reliably converging on these minimal solutions. Chester argues that although *in principle*, one hidden layer is sufficient for any approximation, *in practice* two layers of hidden units may yield better interpolation capabilities with faster training. He notes the problem with using only a single hidden layer is that "the neurons therein interact with each other globally, making it difficult to improve an approximation at one point without worsening it elsewhere." When several layers are allowed, "approximations in different regions can be adjusted independently of each other, much as is done in the Finite Element

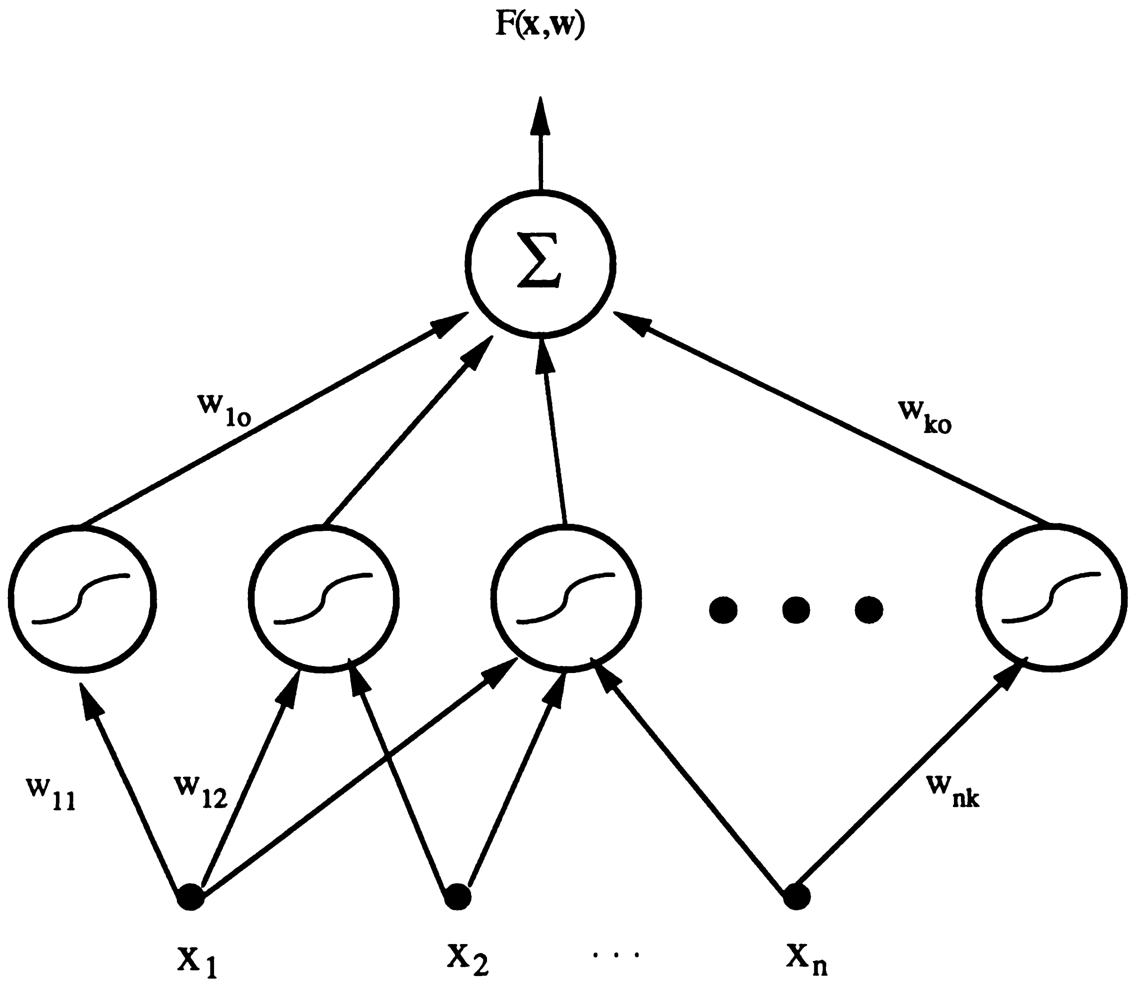


Figure 3.1: Schematic Representation of a Neural Network Applied to Function Approximation

Each output unit of a 2-layer network is a superposition of continuous, nonlinear functions of the input vector $\mathbf{x}$. The output unit is trained to approximate a function $f(\mathbf{x})$ by optimizing the estimate $F(\mathbf{x},\mathbf{w})$ with respect to the weight vector $\mathbf{w}$.

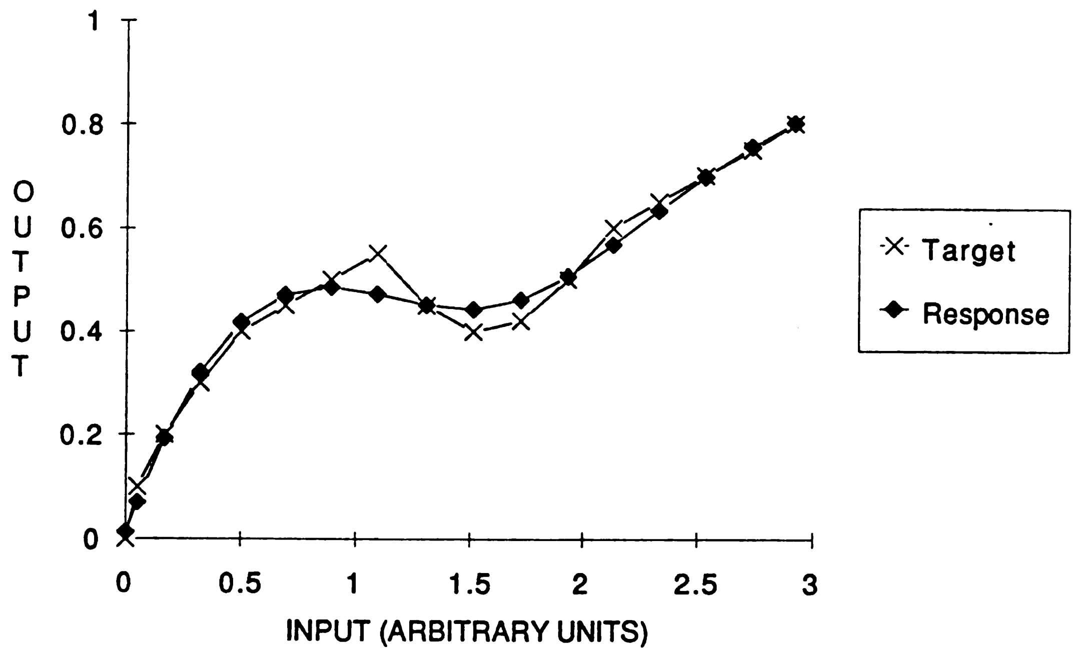


Figure 3.2: Example of Function Approximation

A small neural network (1-15-1) was trained to approximate an arbitrary function of one continuous variable. The network was optimized with the Chemotaxis algorithm until the average squared error per sample in the training set was .01. Training averaged around 1000 cycles through the training set (epochs). The X's denote the patterns used in the training set. The black diamonds represent the trained network's performance. Accuracy of the estimate seems to be related to the sample density and the smoothness of the function being approximated.

Method for solving partial differential equations or the spline technique for fitting curves" (Chester 1990a).

To get an idea of how well neural networks perform this task *in practice*, a network was trained to approximate an arbitrary function of one variable. Figure 3.2 shows the performance of networks trained on an arbitrary function of one variable. Notice that the accuracy of the approximation depends on the sampling density and the smoothness or regularity of the function. In Chapter 6, we exploit the functional approximation properties of neural networks to solve a problem in open-loop motor control.

3.3 PATTERN RECOGNITION

The problem of mathematical pattern recognition can be viewed as a function approximation task. Solution of a problem in pattern recognition involves estimating of a function which indicates a degree of membership in a given class of objects based on observed features. Such a function is called a characteristic function in fuzzy set theory (Zadeh 1968), a discriminant function in the context of pattern recognition, or a probability density function in statistical decision theory (Andrews 1972, pg. 67). The classification problem is to find a multidimensional hypersurface which separates the patterns in pattern space P into the various categories in classification space C which affords some degree of confidence in correctly classifying unknown patterns (Andrews 1972). This mapping is often highly nonlinear, and the methods of finding these surfaces rely on optimization methodologies. Statistical decision theory is the most developed and formalized of the three disciplines. Indeed, when all the relevant statistical information is known, statistical decision theory defines the best theoretical performance of a pattern recognition system.

Neural networks have demonstrated their utility over conventional techniques in ill-posed problems or in problems where not enough classified prototypical patterns are known to apply statistical decision theory. Examples of empirical neural network studies have been presented in pattern recognition problems in vision (Bertero, Poggio and Torre 1988, Lehky and Sejnowski 1988), speech (Lippmann 1989), sonar signals (Gorman and Sejnowski 1988) and seismic signals (Dowla, Taylor and Anderson 1990 & Chapter 4) - among others. This section outlines the subject of pattern recognition with a particular emphasis on statistical decision theory and draw the relevant comparisons to neural networks. Recent theoretical results are presented which argue that, under certain conditions, the methods of neural networks are equivalent to statistical pattern recognition.

3.3.1 FEATURE EXTRACTION

In many problems, pattern space is high or infinite-dimensional as the patterns of interest are taken from measurements of the physical world. The sheer size of pattern space may render a pattern recognition problem theoretically or practically transcomputational (Bremermann 1968, Andrews 1972). Thus, most pattern recognition schemes reduce the problem by choosing some essential "features" of pattern space which are likely to be sufficient for the classification. The problem is schematically illustrated in two parts:

$$\begin{tikzcd} P \arrow[r] & \text{Feature Extraction} \arrow[r] & F \arrow[r] & \text{Decision Algorithm} \arrow[r] & C \\ \text{Pattern Space} & & \text{Feature Space} & & \text{Classification Space} \end{tikzcd}$$

Where classification space C^k consists of a finite number of object classes $\{\omega_1, \omega_2, \ldots, \omega_k\}$.

The primary goal of feature extraction is to reduce the dimensionality of the pattern space. In other cases, however, feature extraction may simply involve translations, rotations or transformations of the pattern space. Proper feature selection, then, may allow simpler decision surfaces between classifications. For example, certain patterns or regularities in temporal signals are more readily identified in the frequency domain. Common examples are problems in speech recognition, sonar signal processing or visual image processing.

In many problems, however, we do not know, *a priori*, what features will be useful. Good examples of this situation are systems where human intuition fails, such as high dimensional systems, or in problems where we have no natural reference, such as what constitutes an interesting olfactory, sonar or seismic pattern. In these cases, it may be more advisable to skip pre-processing entirely so as to not discard potentially useful information. Then feature space is equivalent to pattern space.

Neural networks can be useful in both stages - they can be excellent classification machines and they can be used to find useful features to serve as inputs to more traditional pattern classification techniques (see below and chapter 4).

3.3.2 A BRIEF REVIEW OF STATISTICAL DECISION THEORY

The goal of statistical pattern recognition is to draw decision boundaries - called discriminant functions - between object classes. If we had complete knowledge of the distribution of features in each classification and their relative frequencies, we could calculate the probabilities of membership in each class as a function of the observed values. Since these statistics define an ideal performance limit, it is useful to interpret pattern recognition in the context of statistical decision theory.

To illustrate the pattern recognition problem, we will first consider a simple case where we only have one or two features to discriminate between two classes of objects. While all of the theoretical results presented here can easily be extended to multiple classes and high dimensional feature spaces, it is important to keep in mind the practical difficulties of computing exact solutions to high dimensional problems (for example, inverting matrixes). These limitations often require that simplifying assumptions be made about the distribution of the data and noise, or in other cases, ignore well-established information. We describe empirical means to find discriminant functions, and argue that the function approximation capabilities of neural networks classify them as an alternative non-parametric technique for estimating discriminant functions.

3.3.2.1 THE TWO CATEGORY CASE

Let v_0 represent the observed value of feature vector **x**. Let the prior probability of an object belonging to class ω_i be $P(\omega_i)$ and the conditional probability density function for observation v_0 be $p(v_0|\omega_i)$. [The notational format for this and the remainder of this chapter will represent probabilities as $P(\cdot)$ and probability density functions (p.d.f.'s) as $p(\cdot)$.] The *a posteriori* probability that the observed object belongs to class ω_i is given (by Bayes' Rule) to be:

$$P(\omega_i \mid v_0) = \frac{p(v_0 \mid \omega_i)\, P(\omega_i)}{p(v_0)}$$

where

$$p(v_0) = \sum_{i=1}^{2} p(v_0 \mid \omega_i)\, P(\omega_i)$$

Figure 3.3 shows the conditional probability densities of two classes of objects on a one-dimensional feature x. We can minimize the average probability of classification error using Bayes decision rule (e.g. Duda and Hart 1973):

$$\text{Decide } \omega_1 \text{ if } p(x \mid \omega_1)P(\omega_1) > p(x \mid \omega_2)P(\omega_2)\,; \quad \text{otherwise decide } \omega_2.$$

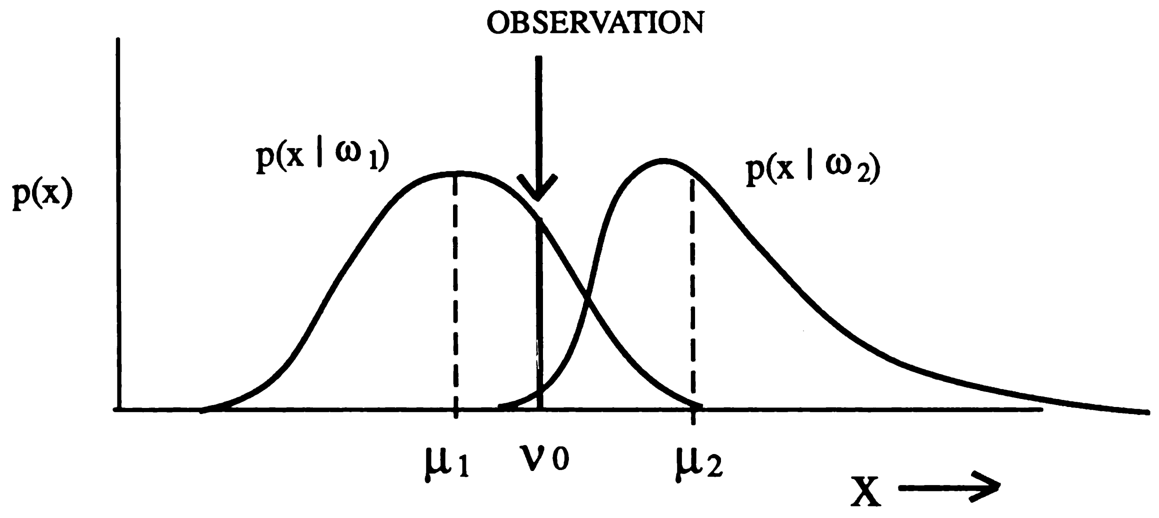


Figure 3.3: Probability densities of two object classes on feature x.

If the prior probabilities are assumed to be equal, then Bayes decision rule becomes:

$$\text{Decide } \omega_1 \text{ if } p(\mathbf{x} \mid \omega_1) > p(\mathbf{x} \mid \omega_2)\,; \quad \text{otherwise decide } \omega_2.$$

the conditional probability density functions are shown where $f_1 = p(v_0|\omega_1), f_2 = p(v_0|\omega_2)$.

3.3.2.2 DISCRIMINANT FUNCTIONS

These probabilities can be used to form the basis of a *discriminant function*, that is, a function which indicates a degree of membership to a given class. For the two-category case, it is convenient to choose a single discriminant which separates the two classes:

$$g(\mathbf{x}) = g_1(\mathbf{x}) - g_2(\mathbf{x})$$

or

$$g(\mathbf{x}) = P(\omega_1 \mid \mathbf{x}) - P(\omega_2 \mid \mathbf{x})$$

In the practical design of neural network applications, however, it is often more convenient to have each output node represent a single discriminant function (see section 3.4).

When working in high dimensional spaces, it is impractical to treat each measurement or feature as an individual discriminant. Part of feature extraction, then, involves combining measurements in pattern space into composite discriminants. The simplest functional form is the linear discriminant function. A discriminant for class k would take the form:

$$g^{(k)}(\mathbf{x}) = w_1^{(k)}x_1 + w_2^{(k)}x_2 + \ldots + w_N^{(k)}x_N = \mathbf{w}^{(k)} \cdot \mathbf{x}$$

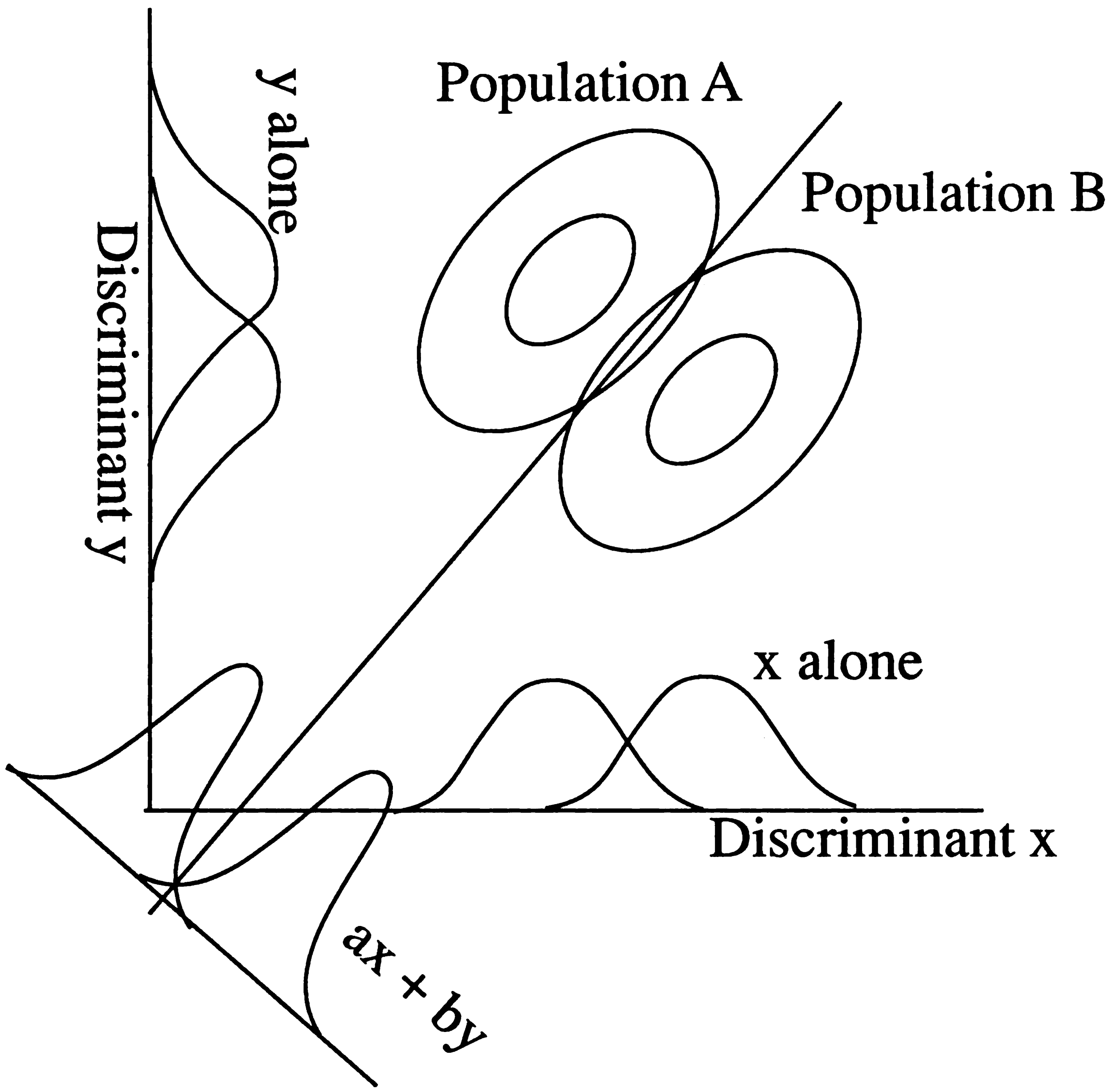


Figure 3.4: Distribution of two object classes on two features

Discrimination between the two classes using either feature alone is not as sharp as an optimized composite, linear discriminant (ax + by). Traditional statistical discrimination theory can find the Bayes optimal discriminant if enough is known about the exact distributions or reasonable assumptions about these distributions can be made. When known patterns are limited, non-parametric "learning machines" can be used to find good separation empirically. In the context of neural networks, the parameters a and b of a linear discriminant represent weights in a single layer perceptron. More complex hypesurfaces can be generated with a multilayer perceptron.

Figure 3.4 shows a *linear* discriminant function for a two-feature problem. The weight vector **w** can be calculated exactly if all the statistical information is known. Otherwise, a practical discriminant function can be found empirically using a variety of methods (e.g. the Fisher discriminant [Duda and Hart 1973, pg.114]). A single layer perceptron can also be trained to find the optimal linear discriminant using the perceptron convergence procedure (Duda and Hart, pg. 141 1973). In chapter 4, we apply both methods to a problem in seismic signal discrimination. Unfortunately, finding nonlinear discriminant functions is much more difficult.

3.3.3 NONPARAMETRIC (DISTRIBUTION-FREE) CLASSIFICATION

When there is not enough data to support the application of statistical decision theory directly, methods of *parametric classification* can be used. "Parametric classification refers to the development of discriminant functions in which the underlying probability functions are assumed *known*. It then remains to simply estimate a set of parameters which will then completely describe the densities of the known prototypes" (Andrews 1972, pg. 113). Usually, data is assumed to be distributed in a Gaussian function. This has two advantages: (1) the Gaussian is the most common distribution in nature, and (2) it allows simplification of the underlying mathematics.

When the form of the probability density functions are not known, decision boundaries must be found empirically. Nonparametric (distribution-free) classification involves learning (from prototypical or "exemplar" patterns). A variety of "learning machines" have been proposed to perform nonparametric classification (Nilsson 1965, Andrews 1972, Duda and Hart 1973). For example, the Widrow-Hoff learning rule and the perceptron convergence procedure are ways to find a linear discriminant function through learning in "weight space". All of these methods incrementally alter the decision surface which separates the prototype patterns. The main advance offered by back-propagation is the ability to draw highly complex decision surfaces.

3.3.4 NEURAL NETWORKS CAN BE USED TO APPROXIMATE THE BAYES-OPTIMAL DISCRIMINANT FUNCTION

Many authors have pointed out that neural network training algorithms can be considered a nonparametric technique for estimating the *a posteriori* probabilities of pattern classifications (Lippmann 1987, White 1989, Wan 1990, Ruck et al. 1990). In fact, recently two independent groups have shown that neural networks are *in theory* capable of learning arbitrarily close approximations of the *a posteriori* probabilities (White 1989; Ruck, et al. 1990). A detailed and simple proof for the two-class and multiple class discrimination problems is given in Ruck et al. (Ruck et al. 1990). Here we present an outline of the argument for the two-class problem. This argument can readily be extended to the multiple class case.

Let **x** represent the feature vector which is to be classified. Consider an arbitrary approximation device that computes an estimate F(**x**,**w**). For a multilayer perceptron, **w** represents the weight vector, although this could apply to any device with adjustable parameters **w**.

The training set consists of a subset S of all possible patterns in classification space (χ). Let X_1 and X_2 represent the set of feature vectors in classes 1 and 2, respectively ($S = X_1 \cup X_2$). If we choose the output unit's target response to be "+1" for class 1 and "-1" for class 2, then the neural network training algorithms (e.g. back-propagation) can be chosen to minimize the error criterion

$$E_S(\mathbf{w}) = \sum_{\mathbf{x} \in X_1} \left(F(\mathbf{x},\mathbf{w}) - 1\right)^2 + \sum_{\mathbf{x} \in X_2} \left(F(\mathbf{x},\mathbf{w}) + 1\right)^2$$

with respect to **w**.

Ruck and colleagues show that as the training set (or sample set) size increases, the Strong Law of Large Numbers implies minimization E_S converges to minimization of the quantity:

$$\varepsilon^2(w) = \int_X [F(x,w) - g_0(x)]^2 p(x)\, dx$$

where $g_0(\mathbf{x})$ is the Bayes optimal discriminant function:

$$g_0(x) = P(\omega_1 \mid x) - P(\omega_2 \mid x)$$

Thus, back-propagation (as well as any training algorithm which minimizes the squared output error) yields a minimum mean squared-error approximation to the Bayes optimal discriminant function. This result is subject only to the mild and reasonable conditions that the training set be sufficiently large to span the feature space in roughly the same proportion as these objects occur in the environment. Hampshire and Pearlmutter (1990) have developed reasonable error measures and additional conditions for an ideal training set.

Several authors have noted that the coding scheme used to represent the target output may be significant in designing successful neural network applications. Clearly, following the construction above, one should choose one output unit for each category in classification space ("0" for low probability, "1" for high probability for each class) if the goal is to estimate *a posteriori* probabilities.

3.3.5 ADVANTAGES OF THE NEURAL NETWORK APPROACH

Many authors have identified the advantages of neural networks over statistical pattern recognition. Not all of these have been rigorously demonstrated. Listed below are the most frequently cited:

1.) Arbitrarily complex decision boundaries can be easily implemented with back-propagation or other training algorithms. Since neural networks can realize highly nonlinear decision boundaries, feature extraction becomes less important (Chen 1990).

2.) "Neural network classifiers are ... non-parametric and make weaker assumptions concerning the shapes of underlying distributions. They may thus prove to be more robust when distributions are generated by non-linear processes and are strongly non-Gaussian" (Lippmann 1987).

3.) Neural networks can perform better if the statistical distribution of the background noise is not available. Performance is much less sensitive to noise or incomplete data than statistical pattern recognition (Chen 1990; Dowla, Taylor and Anderson 1990). (Chen thinks this is due to the distributed nature of the neural network architecture.)

4.) Training a neural network on prototypes directly from pattern space may be useful for discovering useful features. These features may then be used as inputs to more traditional decision theory algorithms.

3.4 GENERALIZATION

3.4.1 BOOLEAN FUNCTIONS

Single layer perceptrons can not solve problems in which the classes can not be separated by a hyperplane (e.g. exclusive "OR"). Rosenblatt and others circumvented some of these problems by adding additional layers of pre-processing consisting of randomly connected perceptrons or standardized feature extraction procedures.

Nevertheless, the failure to generalize the perceptron convergence procedure or the Widrow-Hoff learning rule to multilayer perceptrons presented an insurmountable obstacle to early perceptron research (which was also extremely limited by available computational resources). Thus, the exclusive OR was one of the first problems attacked with *multilayer perceptrons* using the back-propagation algorithm (Rumelhart and McClelland 1986).

Boolean functions, such as parity, represent a kind of "worst case" scenario. There is no reason to expect a neural network to properly generalize a Boolean function when it is trained on a subset of patterns in the corresponding truth table. (For N Boolean variables, there are 2^N corresponding Boolean functions.) In the terminology of classification theory, each of these input/output patterns is potentially a necessary prototype for the training set, since we do not know *a priori* whether it lies on the decision boundary or perhaps defines a separate region in classification space.

3.4.2 LIMITATIONS OF GENERALIZATION

That "trainable networks can generalize" is a widely held belief and the reason for their popularity in applications (Hinton 1989). Examples show that they can indeed perform much better than chance on unseen patterns in classification tasks such as "shape from shading" (Lehky and Sejnowski 1988), "protein secondary structure" (Qain and Sejnowski 1988), and so on. In the training process the networks may generate appropriate categories (Sejnowski and Rosenberg 1987). Some of the examples in the literature are such that precise rules are not even known (for example for protein secondary structure). On the other hand there are problems where the training set cannot possibly determine the performance on the unseen patterns. For example, there are 2^N Boolean functions for N-bit input strings, and the predictability of the output for "unseen" inputs from those already seen is no better than pure chance.

In a series of papers, Van den Broeck and Kawai (1990; 91) have investigated the generalization properties of Boolean networks. Boolean networks consist of a set of Boolean gates (such as AND, OR, XOR, etc.) linked up in a feed-forward configuration. Study of these binary networks is simpler than that of continuous networks, since the total number of configurations is finite for a given number of gates. When randomly wired, Boolean networks have a bias toward certain Boolean functions. They show that the volume of phase space (number of configurations which implement a particular function) occupied by a given function is widely variable. (When functions are ranked according to frequency of occurrence, volume in phase space for lower ranked function falls with the characteristic inverse power law known as Zipf's law, seen in 1/f noise.) Thus, Boolean networks have an inherent structure of hierarchical hypotheses. The complexity of learning a given Boolean function is related to the number of possible network configurations that solve the problem.

Standard, feed-forward neural networks likewise have a bias towards particular hypotheses about the classification function to be implemented. Generalization in a feed-forward network, therefore, depends on whether the classification function the user is looking for is among one of the more "reasonable" hypotheses from the network's point-of-view. Training time will likely be related to the Hamming distance from the initial guess as well.

Wilson (1991) compares generalization of networks trained by a variety of training algorithms - including the Chemotaxis algorithm. He claims that no single training algorithm can guarantee consistently better generalization performance. The most important factor in generating good generalization, as in functional approximation and statistical pattern recognition, is that the training set form a basis set over pattern space. Predictability, thus, depends not only upon the network structure and the training algorithm but upon the type of association to be learned.

3.4.3 A SIMULATION EXAMPLE

The generalization capabilities of a network cannot reliably be predicted from theoretical considerations. However, one might expect generalization in special cases, where the classes are well represented in the training set. The identity map is geometrically a very special case which may be expected to be easily generalized.

Crick (1989) emphasizes the importance of generalization and categorization in the seemingly trivial problem of stacking networks one on top of each other and requiring that the output equals the input (i.e. the encoder problem). The encoder problem is to retain an identity mapping though several intermediate stages of processing. When trained with an algorithm, starting with random weights, this problem is not trivial at all. In the following, we explore the generalization capabilities of networks trained with the chemotaxis algorithm with a computer simulation on the encoder problem.

Note that there are 16 possible binary input vectors. For the performance tests we had trained with (1,0,0,0), (0,1,0,0), (0,0,1,0), and (0,0,0,1). If outputs for the remaining vectors were random, there would be a 1 in 12 chance for the correct output to occur. Such an "all-or-nothing" comparison, however, does no justice to output vectors that are "almost correct", say off by one bit. In the brain input and output could be "pixels" of an image, for example on the retina and in the visual cortex. Scoring an image where just one or a few pixels are wrong as totally wrong (the same as two images with little or nor correlation) would make no sense. We therefore used the "Hamming distance", the number of bit-by-bit discrepancies between input and output, as measure of similarity (see Table 3.1 and Figure 3.5).

We found that generalization was weak when the training set consisted of only four vectors. We thus increased the training set to eight vectors and found that these nets do indeed "generalize".

Table 3.1: Training Sets Used in the Generalization Study. *

Decimal	Input Pattern	Training Set		
		#1	#2	#3
0	0 0 0 0		X	X
1	0 0 0 1			X
2	0 0 1 0	X		X
4	0 1 0 0	X		X
8	1 0 0 0			X
3	0 0 1 1			X
5	0 1 0 1	X	X	X
6	0 1 1 0			
9	1 0 0 1	X	X	X
10	1 0 1 0	X	X	
12	1 1 0 0	X	X	
7	0 1 1 1	X	X	
14	1 1 1 0	X		
13	1 1 0 1			
11	1 0 1 1		X	
15	1 1 1 1		X	

*Three training sets were used to investigate the property of generalization in the encoder problem. Training sets numbered '1' and '2' were found by random selection of the sample input patterns. Training set #3 consists of patterns with a Hamming distance of 1 of either (0,0,0,0) or (0,0,0,1).

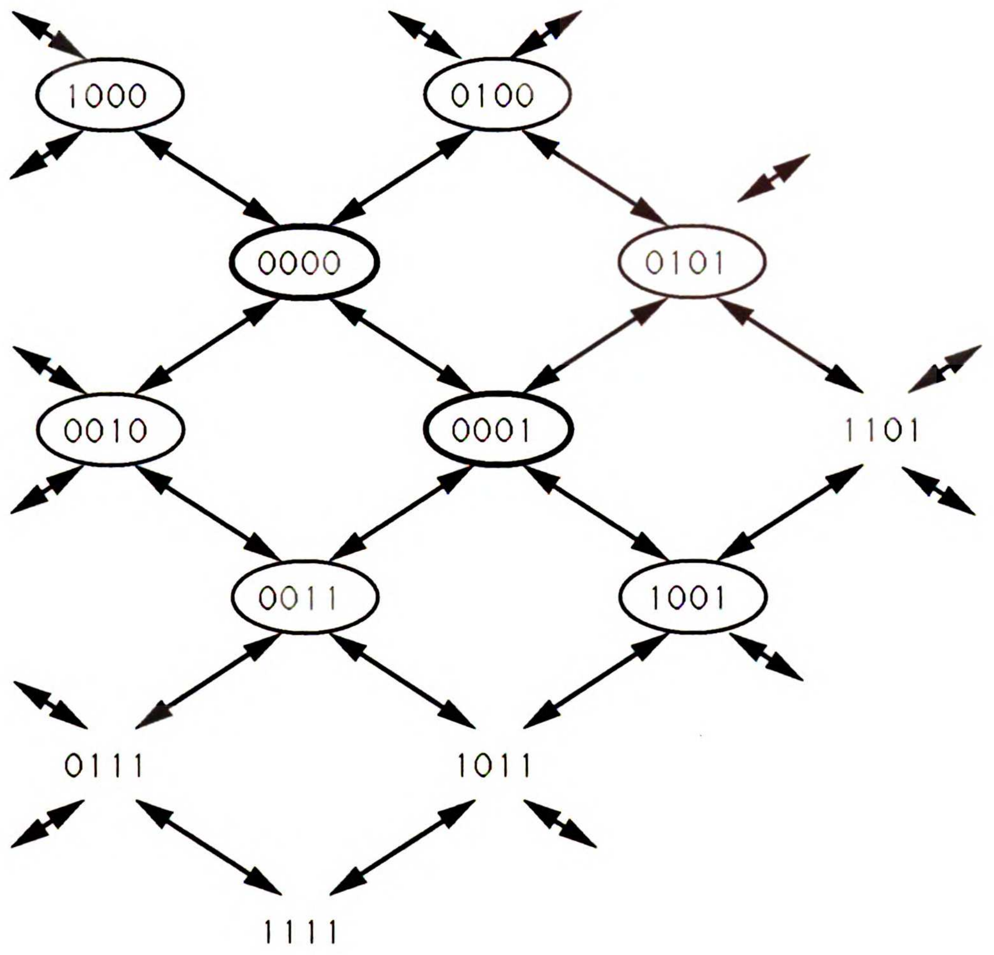


Figure 3.5: 4-Bit Hamming Space

Strings of 4 binary bits have 16 possible configurations. Each 4-bit string has four neighbors with a Hamming distance of 1 (denoted by two-way arrows). Training sets 1 and 2 represent random samples in Hamming space. Training set #3 consists of the 8 patterns in the neighborhood of '0000' and '0001' (circled), which is rather localized in Hamming space.

A training set consisted of 8 of the 16 possible binary input vectors. The remaining 8 patterns comprised the 'test' set. Networks of varying architectures were trained on three different training sets (Table 3.1). Sets #1 and 2 were selected randomly and appear evenly distributed in Hamming space. Set #3 was chosen to be particularly self-similar. Every vector in set #3 is within a Hamming distance of 1 from either the vector (0,0,0,0) or the vector (0,0,0,1); while every vector in the corresponding test set has a Hamming distance of 2 or more from these two vectors (see Figure 3.5). It was anticipated that set #3 would be easy to learn (since the vectors are close) but might not generalize well when tested on the remaining vectors (which are distant).

The networks were trained with the chemotaxis algorithm. Performance was measured by the number of bit-to-bit matches from the input pattern to the output patterns. Thus, for eight 4-bit input vectors there are a total of 32 bit matches possible. A bit match was counted for every bit which was closer to the desired output than 0.5 (squared error of 0.25). Training was continued until the sum of the squared errors over all 32 bits was below 0.3 or for 1800 cycles - whichever came first.

Performance for the three training sets on increasingly restrictive architectures are given in Table 3.2. As can be seen in the table the 4-2-4 nets can perform reasonably well on the 8 patterns in the training set but does not generalize well. For nets with only one extra hidden unit generalization improves significantly but judicious choice of the test set is important.

Table 3.2: Generalization as a Function of Training Set and Net Architecture for the Encoder Problem.*

	4-4-4 Net		4-3-4 Net		4-2-4 Net	
Set	Training	Test	Training	Test	Training	Test
#1	88%	75-100%	88%	72-100%	78-84%	63-78%
#2	100%	69-75%	100%	63-72%	91-97%	56-69%
#3	97-100%	41-59%	100%	38-56%	84-97%	44-50%

*Performance was measured by the number of bit-to-bit matches from the input patterns to the output patterns divided by N=4 times the number of vectors tested. Note that generalization (to the test set) depend on the choice of the training set. Note also the overall degradation of performance when an insufficeint number of hidden units are provided (4-2-4). As anticipated, set #3 was easier to learn (due to the pattern's similarity) but did not generalize well when tested on the remaining vectors.

3.4.4 DISCUSSION: GENERALIZATION, CORTICAL MAPS AND LEARNING RULES

The encoder problem is that of maintaining an identity map through several intermediate processing layers. The importance of such mappings is obvious, since analogous mappings are common in the somatosensory and visual cortex. These mappings are continuously being established, maintained and refined (Chapter 2; Merzenich et al. 1987, 1988).

The identity map (abstracted by the encoder problem) is mathematically and geometrically trivial. Thus, it is relatively easy to generate such mappings using widely varied models and learning mechanisms. For example, an identity map can be created assuming only local excitation and long-range inhibition in a network of processing nodes (Kohonen 1984). Hebbian learning rules can also account for cortical mappings (Grajski and Merzenich 1990). We have shown here that, assuming a representative training set, the chemotaxis algorithm is capable of training a *randomly*-connected network to solve the problem as well. Neurotropic factors are likely also involved (Molnar and Blakemore 1991). Elements of all of these mechanisms may be responsible for maintenance of cortical maps. Hence, one cannot distinguish between learning rules using the encoder problem as a benchmark.

CHAPTER 4

AN EXAMPLE OF PATTERN DISCRIMINATION WITH NEURAL NETWORKS ON SEISMIC DATA[1]

4.1 MOTIVATION

4.1.1 THE REAL WORLD OF PATTERN RECOGNITION

Too often, neural network algorithms are only tested on easy "toy problems" or on benchmark problems (such as the encoder and N-bit parity problems) that are far removed from more "natural" pattern recognition tasks.

"Real-world" problems (such as the recognition of hand-written characters or human speech) commonly have high-dimensional pattern or classification spaces and continuously valued variables. Furthermore, many problems, such as determination of three dimensional structure from a two-dimensional image or stereo vision are "ill-posed" (Poggio and Edelman 1990; Bertero, Poggio and Torre 1988; Geman and Geman 1988).

In this chapter, we present an example of the power of neural networks as pattern classifiers on a "real-world" problem in seismic signal classification. Although this is not a "biological" problem specifically, one pattern recognition problem is as good as any other for the purpose of demonstration. Seismic signal discrimination has several features common to biological pattern recognition problems in vision or acoustics.

---

[1] Results of this chapter appear in Dowla, Taylor and Anderson (1990).

4.1.2 NEURAL NETWORKS VERSUS STATISTICAL METHODS FOR DISCRIMINATION OF SEISMIC SIGNALS

This problem is typical of many in pattern recognition and it also provides a valuable comparison of neural network methods over statistical pattern recognition:

(1) Little is known about the underlying physics and geology of seismic wave sources and propagation. Thus, the form of a seismic signal cannot be deduced from first principles. Selection of useful features or discriminants, therefore, is based on little else than intelligent guesswork (see Sections 4.2.4 & 4.4). Potentially, neural networks can be used to find useful discriminants *empirically*.

(2) No human "expert" can reliably classify a signal even through detailed analysis. (The only way to know for sure whether a seismic signal was produced by a small magnitude underground nuclear explosion is to set one off.)

(3) The pattern space is high dimensional which presents computational problems for traditional statistical pattern recognition (as well as human intuition). Traditional techniques also require the inversion of large dimensional and (potentially) poorly conditioned matrices.

(4) Because of the limited data set and the large number of variables, the statistical distributions of both classes of signals and background noise are unknown. Application of statistical methods, in this case, requires that we *assume* a distribution (usually Gaussian) for the data. As we discussed in Chapter 3, if the real distributions are significantly different from those assumed, performance of statistical classification is degraded.

4.1.3 AN ENGINEERING APPLICATION OF A BIOLOGICALLY INSPIRED COMPUTATIONAL METHOD

In this chapter, we demonstrate the effectiveness of both the chemotaxis and back-propagation algorithms on training feed-forward networks on a non-trivial task. We show that using even simple feed-forward neural network architectures (82-1-2), high discrimination performance can be achieved. We found that neural networks were not only useful in drawing effective decision boundaries but in choosing the discriminants as well. Performance of networks trained by the chemotaxis algorithm was equal to those trained with back-propagation. For ease of implementation, however, we used an automated back-propagation routine (SunNet) to obtain performance statistics on the over 800 simulation runs. Performance of our preliminary study, using a single-layer perceptron, proved to be slightly better than a Fisher discriminant applied to the same task. Subsequent work with an expanded input database using multilayer networks proved to be superior to linear regression techniques (Dowla 1991, in preparation).

There is one final problem involved in tackling a real world problem of this kind, the real-world practical requirements: funding, computer resources, access to real data and expert collaborators. Working at the Lawrence Livermore National Laboratory (LLNL) offered all four.

4.2 BACKGROUND: SEISMIC SIGNAL DISCRIMINATION

4.2.1 NUCLEAR TEST BAN TREATIES

Discrimination of seismic records from natural earthquakes and underground nuclear explosions is a difficult and important problem in test ban treaty verification research (Dahlman and Isrealson 1977). The eventual goal of these studies is to monitor and verify compliance with a variety of potential treaties ranging from a

Threshold Test Ban Treaty (TTBT) to a Comprehensive Test Ban Treaty (CTBT) (Taylor, March 1990). Of particular interest is the ability to discriminate low magnitude ($m_b < 4$) earthquakes from underground nuclear explosions at regional distances ($\leq$ 2000 km). This would allow for verification with a small network of seismic stations spread across each signatory nation.

4.2.2 CHARACTERISTICS OF SEISMIC DATA

Both earthquakes and underground explosions generate seismic waves which travel through the earth (Body waves) and waves which propagate along the surface (surface waves). There are two main types of surface waves: compressional waves (P waves) and shear waves (S waves). A Rayleigh is one type of surface wave, that appears on the vertical- and radial-component seismograms. A Love wave is another type of surface wave, with particle motion perpendicular to the wave's horizontal direction. The Lg phase is also a surface wave. It is a higher-mode wave that is guided by the crust of the earth and is thought to be a superposition of higher-mode Rayleigh and Love waves (Taylor, March 1990). The L_g phase is often the largest in amplitude.

Differences in the propagation medium and regional geology affect the attenuation and arrival time of various phases of a seismogram recorded at a seismic recording station. For example, in the Great Basin in Nevada, the most prominent regional phases are the L_g, P_g and P_n waves. Figure 4.1 shows typical seismograms for a large magnitude explosion and earthquake, passed through standardized bandpass filters. Compressional phases (P and P_g) arrive first, followed by the slower shear phases (S, L_g, Rayleigh and Love waves).

4.2.3 EARTHQUAKES VERSUS EXPLOSIONS

Most discriminants are chosen based on physical differences expected between earthquakes and explosions. Generally, explosions

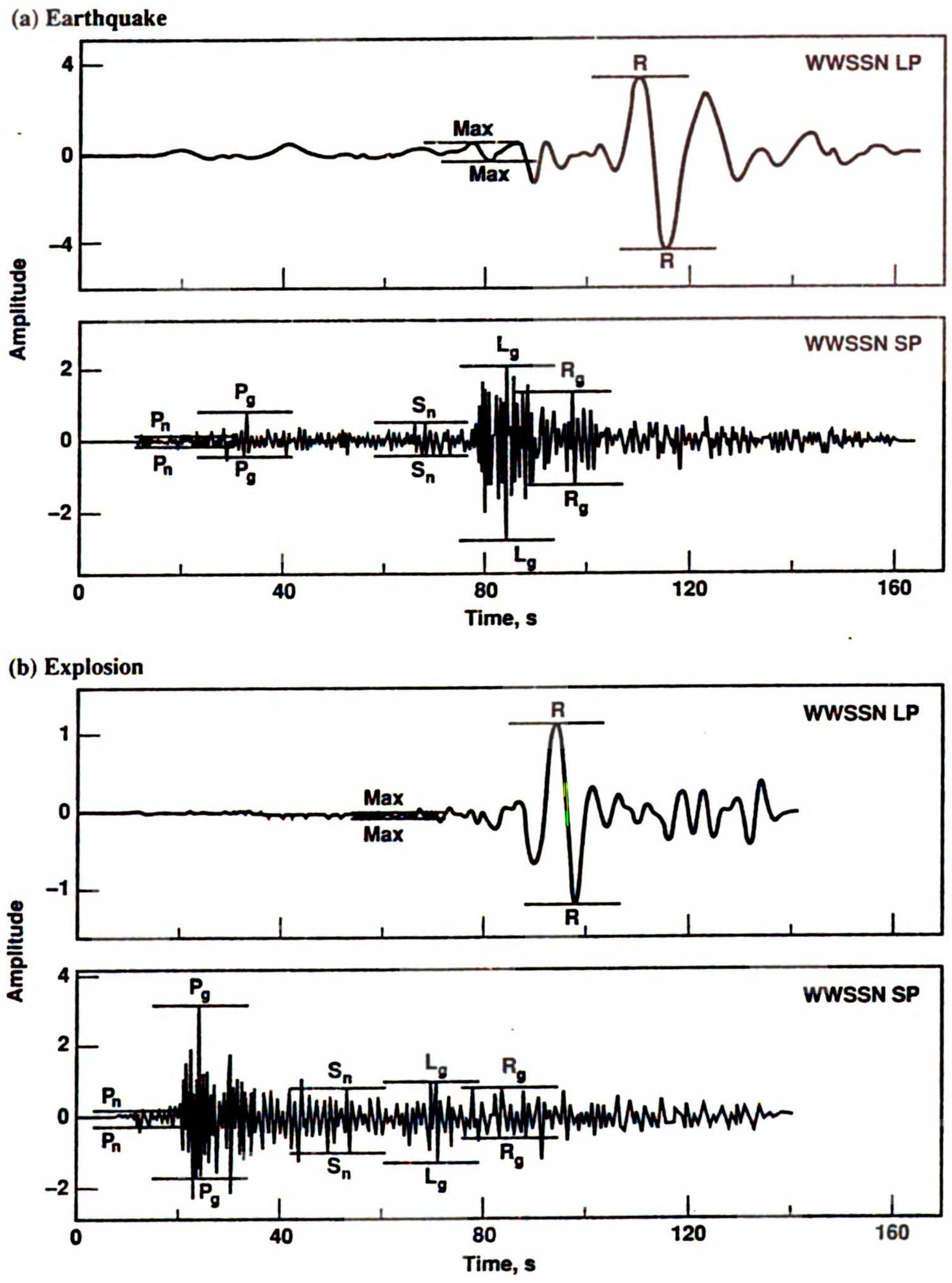


Figure 4.1: Examples of peak-to-peak earth-motion amplitude measurements for a magnitude-5.7 earthquake (a) and a magnitude-6.0 explosion (b) passed through WWSSN (World-Wide Standardized Seismic Network) LP (Long Period time-domain amplitudes) and SP (Short Period) bands. The onset of the various wave signals can be determined from these signals. (Figure from Taylor [March 1990]).

can be considered a source of *compressional* waves radiating uniformly from the explosion site; although they can trigger the release of shear energy in nearby faults. Earthquakes are the result of a release of elastic energy in a fault, and therefore contain a relatively larger component of the release energy in shear waves. Explosions are also a sudden event, generating high frequency waves. Earthquakes are often a 'deep' source, with the epicenter located deep within the crust. Taken together, we would expect that for the same magnitude event, the seismic signal of explosions would contain a larger percentage of energy in surface and compressional waves and in higher frequencies.

4.2.4 DISCRIMINATION STUDIES

Currently, discrimination of regional data is an important research topic, and a variety of regional discriminants have been proposed (cf. Pomeroy et al. 1982, Taylor et al. 1989). Discrimination of small magnitude events, however, is still a difficult problem. For small magnitude events (body-wave magnitude = $m_b < 4$), spectral discrimination using multiple regional phases has recently received much attention (Bennett and Murphy 1986, Taylor et al. 1988). It is generally believed that both spectral shapes and ratios of the regional (P_n, P_g and L_g) might be quite useful for distinguishing earthquakes and explosions (Pomeroy et al. 1982). Generalization and regionalization of these discriminants is, however, important for optimum performance. Taylor, Sherman and Denny (1988) evaluated the effectiveness of several discriminants based on L_g and L_g/P_g ratios. Figure 4.2 shows the performance of some of these discriminants on real data (Taylor, March 1990). Note that discrimination tends to be more precise for larger amplitude events.

Since neural networks can classify populations by generating complex discriminant functions by training on real data, we used as input to the network the full broadband, distance-corrected spectra of the regional seismic phases. During the learning phase, the

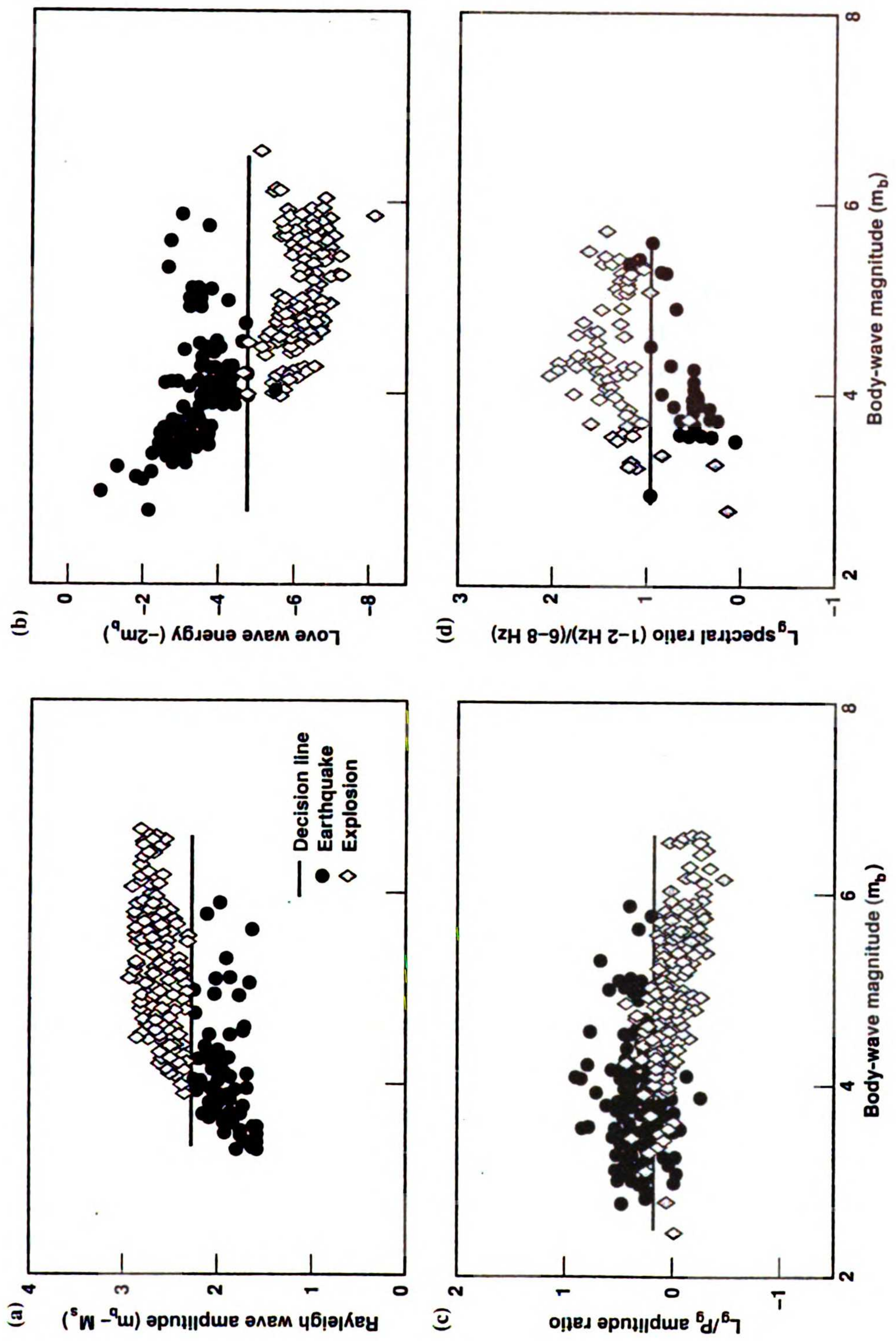


Figure 4.2: Network-averaged discriminants plotted versus body-wave (m_b) magnitudes: (a) Rayleigh wave amplitude, (b) Love wave energy, (c) L_g/P_g amplitude ratio, and (d) L_g spectral ratio. (Figure from Taylor [March 1990]).

network automatically extracted and learned the relationships among the discrete frequency components of the multiple regional phases for correct discrimination between earthquakes and explosions. The network was developed and tested with a large number of real seismic events, consisting of 83 earthquakes and 87 underground nuclear explosions recorded at each station in a network of four stations located in the western United States. Results of this study based on regional spectral data indicate that neural networks can indeed generate excellent discriminant functions. The rate of correct recognition for untrained data is over 93 percent at any single station and is 97 percent for a network of four stations.

Our primary goal at the outset of this study was to gain an understanding of the performance of neural networks for seismic event discrimination with a set of real seismic data. In this chapter, we also discuss the important problems of the representation, pre-processing, normalization, and training of neural networks with a database of real seismic data.

We discuss the problems of pre-processing and data representation in neural networks. This is followed by a discussion of the seismic spectral data for discrimination and the performance of the network for discrimination between earthquakes and explosions. We then apply the same data to the conventional Fisher discriminant (Tjøstheim 1981, Duda and Hart 1973), a linear method which utilizes covariance matrix information, and compare its performance with that of the network. Finally, we conclude with a discussion of the implications of our results and ares of future research in seismic neural networks.

4.3 A NEURAL NETWORK APPROACH

Although in the seismic discrimination problem we used a multilayer perceptron, the architecture of the network which we eventually used had the structure of a very simple network with only one hidden unit and two output units. We found that for this data set, increasing the number of layers or hidden units did not improve the performance significantly. In order to gain insight into the network, we chose to keep the architecture to a minimum level of complexity.

To gain further insight into the learning method consider an explosion spectrum presented at the input of the network shown in Figure 4.3. (In neural network terminology, this network has just one hidden unit with N units at the input layer and 2 units at the output layer.) Input to the network are spectral values, i.e, in this problem the network input layer units corresponds to spectral frequency components of the three phases. The input layer is thus represented by a vector of (N = 3M) elements:

$$\mathbf{x} = [S_{Lg}(1),\ldots,S_{Lg}(M), S_{Pg}(1),\ldots,S_{Pg}(M),S_{Pn}(1),\ldots,S_{Pn}(M)]$$

corresponding to the (M = 41) frequency components for each of the three phases, sampled logarithmically from 0.1 to 10 Hz. The mechanics of how we computed the spectra is described in the next section.

In order to teach the network to discriminate between earthquakes and explosions, a training set consisting of a large number of spectra for both earthquakes and explosions was applied to the network. The back-propagation algorithm was then used on the network where the objective was to adjust and determine the weights **w** such that for each member of the training set the network classified the spectra according to its appropriate category: an output of 1 for explosions and an output 0 for earthquakes. When the network was tested, for example, an activation level of around 0.5 means the network is unable to decide on the event type.

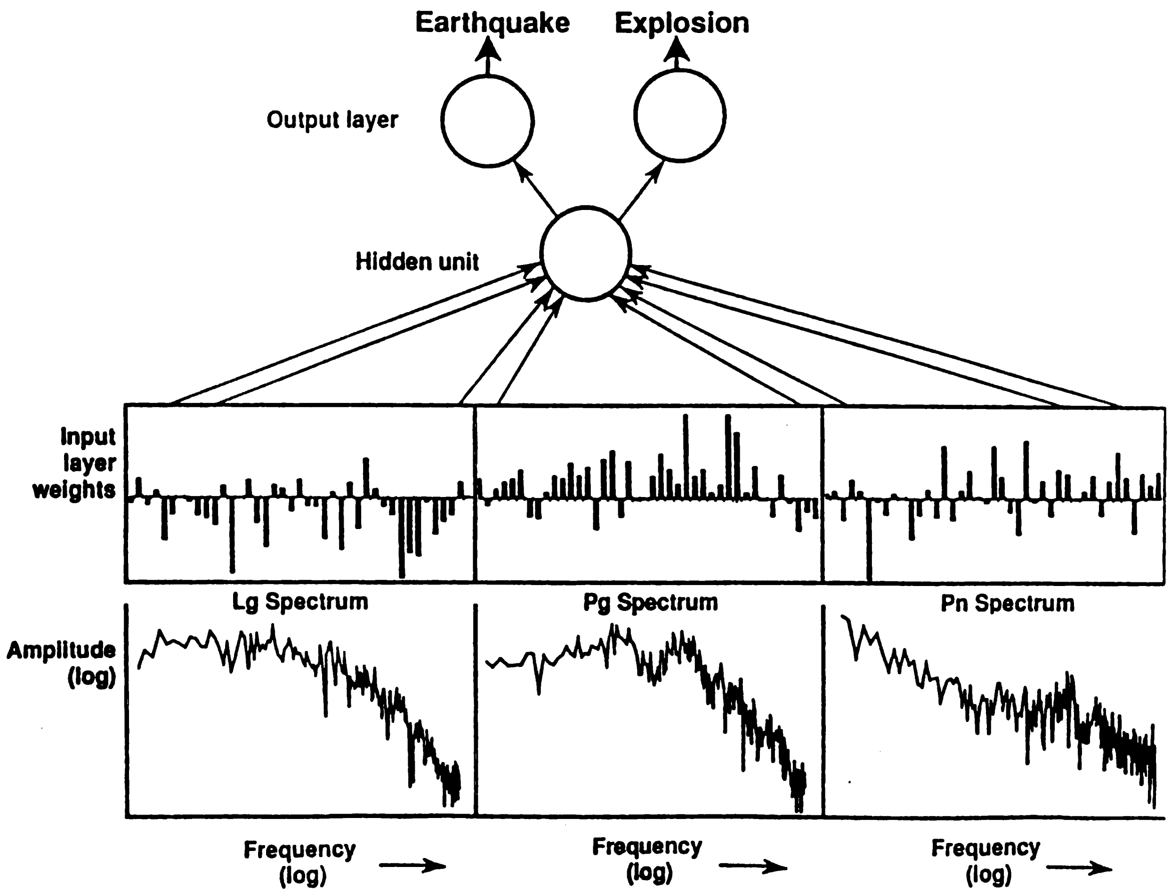


Figure 4.3: The architecture of a neural net which was used to discriminate between explosion and earthquake spectra. The network had 123 input units (corresponding to the 41 spectral values of the three phases), 1 hidden unit, and 2 output units. The back-propagation training algorithm was used to teach the network to discriminate between the explosion and earthquake spectra at the input.

4.4 SEISMIC SPECTRAL DISCRIMINANTS

4.4.1 CHARACTERISTICS OF THE SPECTRAL DATA

Examples of regional seismic phases from earthquakes or underground nuclear explosions which are recorded by the LLNL seismic observatory stations are illustrated in Figure 4.4. For the Basin and Range region, the regional phases P_n, P_g and L_g are the principle arrivals corresponding to distinct modes and paths of seismic wave propagation. The first arrival in our data is usually P_n, a body wave with longitudinal particle motion. P_n is usually followed by P_g, also a longitudinal wave with frequencies slightly higher than P_n . However, for the instruments we used the range of frequencies for both P_n and P_g were from 0.1 to 10 Hz. In this geologic region P_g is observed to have higher amplitudes than P_n . The final principle arrival is L_g , a regional phase whose properties are not fully understood in spite of the fact that the L_g usually has the largest amplitude of the three phases. L_g has frequencies approximately in the same range as that of P_n and P_g . It is generally believed that the L_g phase is a superposition of higher modes of Love and Rayleigh waves propagating in a crustal wave guide. Typical amplitude spectra of the principle regional phases are illustrated in Figure 4.5.

In our notation we represent the amplitude spectrum of a phase at frequency f_k by $S_{phase}(f_k)$. For example, for the L_g phase, the amplitude spectrum will be represented by $S_{Lg}(f_k)$, the magnitude of the discrete Fourier transform of $s_{Lg}(n)$:

$$S_{Lg}(f_k) = \left|\sum_{n=0}^{N-1} w(n) s_{Lg}(n) e^{-j2\pi f_k n \Delta t}\right|$$

where $s_{Lg}(n)$ represents the discrete-time windowed sequence corresponding to the phase L_g, $w(n)$ represents a cosine tapering

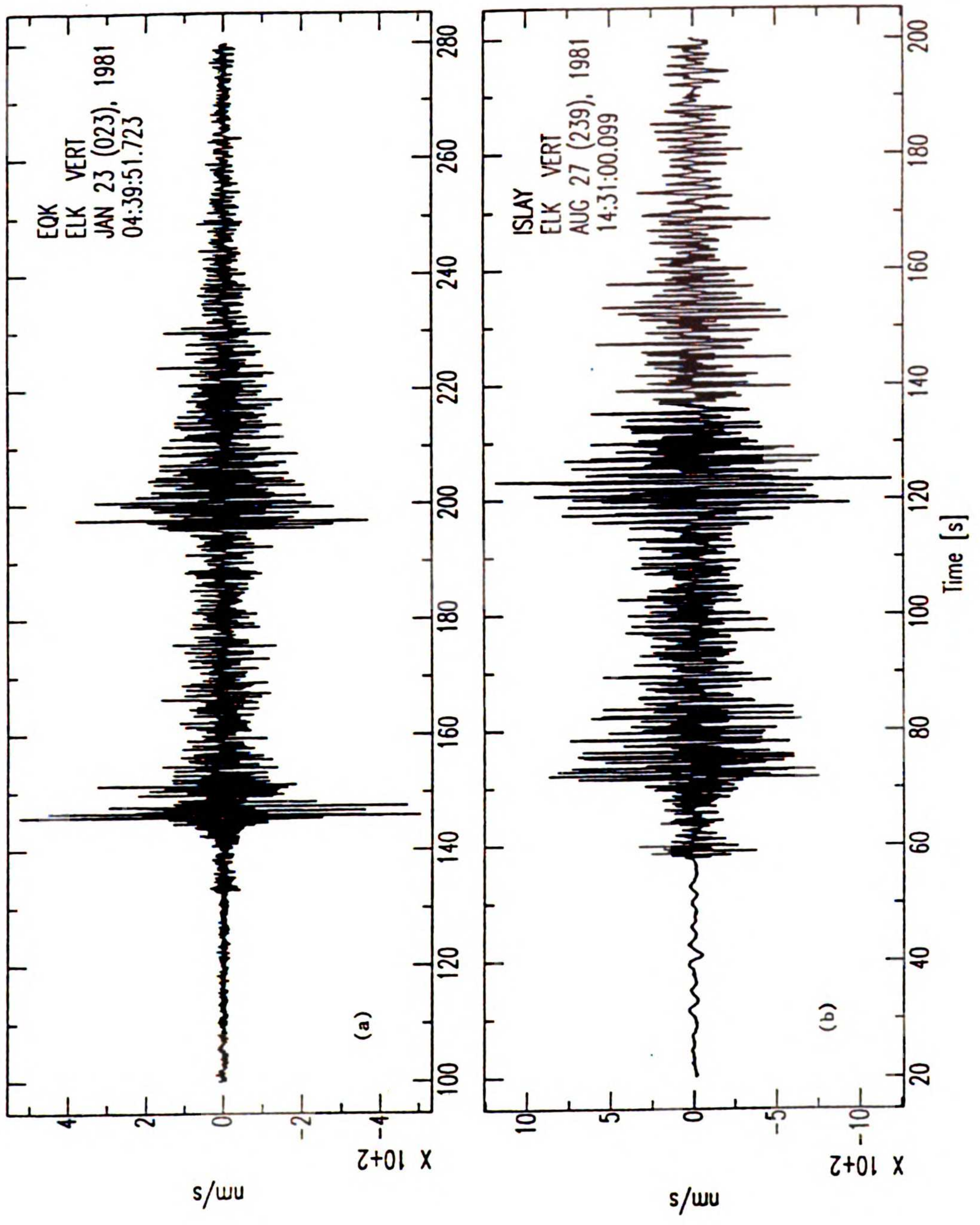


Figure 4.4: Examples of seismograms for (a) an earthquake and (b) an underground nuclear explosion recorded at Elko, NV, one of the four stations of a seismic network operated by LLNL.

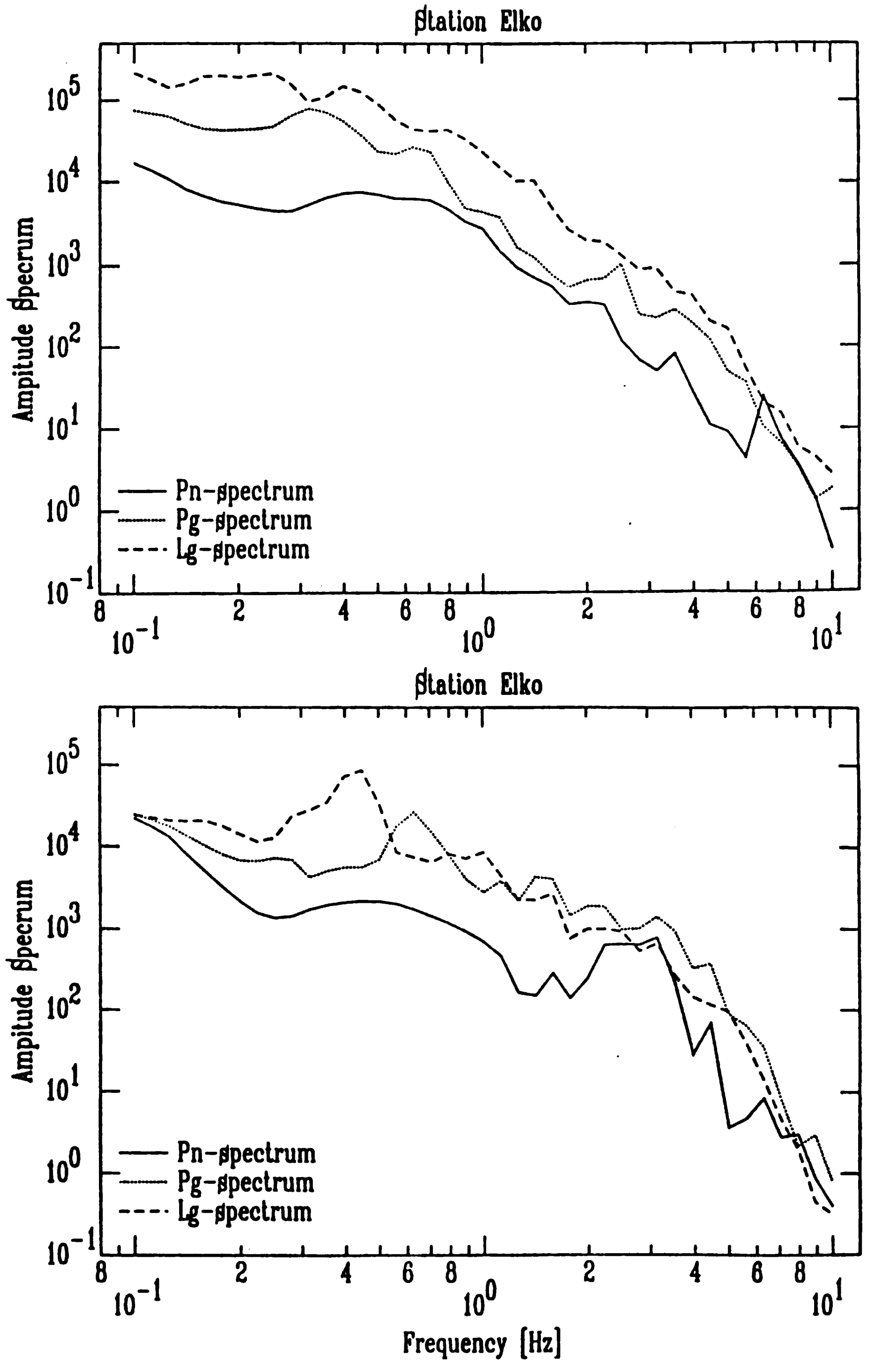


Figure 4.5: Examples of displacement amplitude spectra (nm/Hz) of the P_n, P_g and L_g phases of an earthquake (a) and an underground nuclear explosion (b) recorded at Elko. Note that the smooth characteristics of the spectral plots is due to the fact that the spectra were sampled in a log scale and plotted with linear interpolation between sampled points.

window, and Δt represents the sampling interval. For most of the data in this study, Δt is 0.025 seconds.

With these brief introductory comments on the seismic waves observed at the LLNL network, we state more precisely the seismic event discrimination problem which we studied:

Given a seismogram with detected regional phases L_g, P_g and P_n, extract and use the characteristics of these phases to determine whether the source responsible for generating these waves was an earthquake or an underground nuclear explosion.

4.4.2 SEISMIC DISCRIMINANT FUNCTIONS

In a review paper addressing the above problem, Pomeroy et al. (1982) summarize 15 classes of regional discriminants that could be used in the discrimination problem. They concluded that while these discriminants had differing degrees of success in separating earthquakes from explosions, more research was required to determine clearly the most promising discriminants. They give an extensive discussion on the L_g/P amplitude ratio discriminant and conclude that while L_g/P is a promising discriminant, earthquakes and explosions often, based on this discriminant overlap significantly. A number of studies have extended these time-domain discriminants into the spectral domain and have conducted systematic comparisons on the performance of various discriminants. In a recent study, Taylor et al. (1988) extended the spectral ratio discriminant of Bennett and Murphy (1986) to higher frequency bands for the same phase. They define the L_g phase a spectral ratio:

$$D(L_g) = \sum_{k=K_{11}}^{K_{12}} S_{Lg}(f_k) \Big/ \sum_{k=K_{21}}^{K_{22}} S_{Lg}(f_k)$$

where (K_{11}, K_{12}) and (K_{21}, K_{22}) define upper and lower frequency bands of the spectra. They found that for a certain threshold level, a low value for D(Lg) indicates a waveform was due to an earthquake. On the other hand, a high value for D(Lg) would indicate an explosion. by applying their discriminants on real data, Taylor et al. concluded that spectral ratios at certain frequency bands ([K_{11}, K_{12}), [K_{21}, K_{22}]) of a phase might be an important discriminant. Their results indicate that L_g, in comparison with P_g and P_n, performs the best for single-phase event discrimination. Performance in terms of misclassification probabilities ranged from 4 to 33 percent, depending on the phase and the station, and from 7 to 16 percent for the network of four stations of the LLNL Nevada Test Site (NTS) stations. In summary, the study by Taylor et al. (1988) shows clearly that the broadband spectral characteristics of the regional phases might be quite important for the discrimination of regional seismic events.

In view of the studies of Bennett and Murphy (1986) and Taylor et al. (1988) on spectral discriminants, it is reasonable to guess that an optimum discriminant might be a weighted spectral ratio of the form

$$D(L_g) = \sum_{all\,k} W_1(f_k) S_{Lg}(f_k) \Big/ \sum_{all\,k} W_2(f_k) S_{Lg}(f_k)$$

where $W_1(f_k)$ and $W_2(f_k)$ are spectral weightings at frequency f_k. In the previous discriminant, W(k) = 1 for i = 1,2. Another discriminant might be of the form of a ratio of spectral components of the different phases, or a *multi-phase spectral ratio*:

$$D(L_g, P_g) = \sum_{all\,k} W_{Lg}(f_k) S_{Lg}(f_k) \Big/ \sum_{all\,k} W_{Pg}(f_k) S_{Pg}(f_k)$$

In fact, the discriminant function need not be ratios but could conceivably be some unknown function of the various frequency components of the three principle phases. In any event, the problem reduces to one of determining these unknown relationships and the optimum weighting functions $W_x(f_k)$. Given the large number of unknown variables like geology and source characteristics, analytical solution of this problem proves to be difficult.

The application of a "learning machine", like a neural network, which determines discriminant functions automatically by systematically training on real data might be an alternative method for constructing the discriminant function. In this study we focus on spectral discrimination using artificial neural networks using the frequency components of the principle seismic phases.

4.5 SEISMIC DATABASE AND PREPROCESSING

The results of this study are based on data from the four LLNL seismic stations at Elko (NV), Kanab (UT), Landers (CA), and Mina (NV), located at distances of 200 to 400 km from the Nevada Test Site (NTS). The database consisted of 83 earthquakes and 87 nuclear explosions at each of the four stations of the LLNL NTS network. All the explosions occured at the Nevada Test Site. Epicenters of the earthquakes and the relative locations of the stations are shown in Figure 4.6.

In order that discrimination is based on the basis of source type, and not unduly influenced by background noise, event distance or event magnitude, for each event in the database we checked the signal to noise ratio, performed a distance correction and a magnitude normalization on the spectral data as routine pre-processing steps. We explain the pre-processing of the data in following sections.

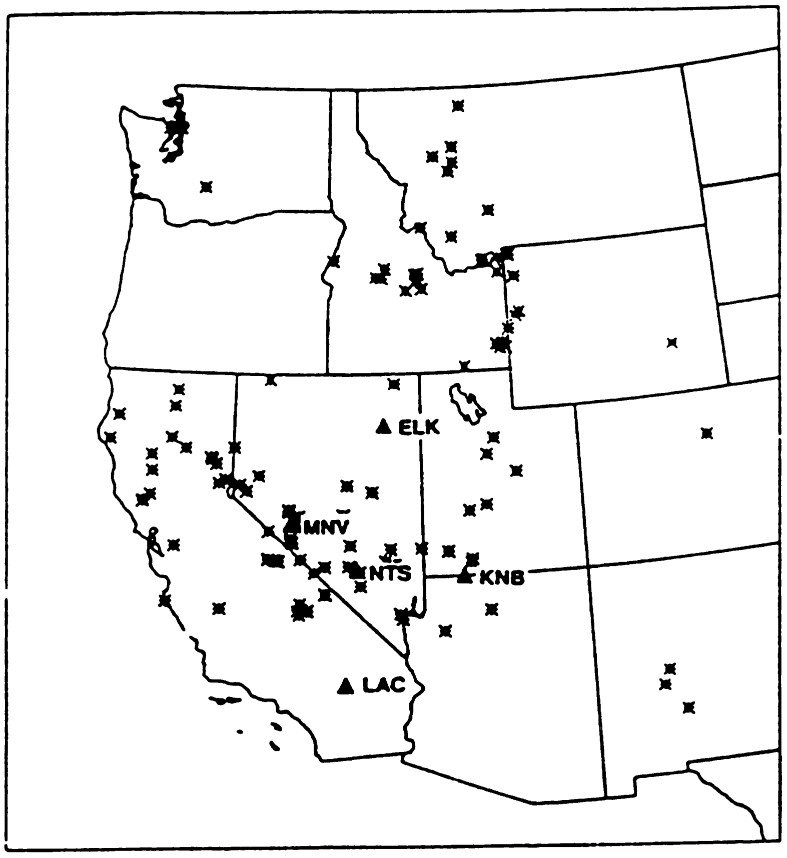


Figure 4.6: Map of the LLNL Nevada Test Site (NTS) seismic network. The network consists of four stations, indicated by the filled-triangles, located 200-400 km from the Nevada Test Site. The locations of the earthquake epicenters used in this study are indicated by asterisks.

4.5.1 SPECTRAL ESTIMATION AND SIGNAL-TO-NOISE RATIO CHECKING

For each event-station pair, the spectra were calculated from windowed P_n, P_g and L_g phases. Group velocity windows were defined by t_1 and t_2, were $t_1 = \Delta/6.0$ and $t_2 = \Delta/5.0$ for the P_g phase, and $t_1 = \Delta/3.6$ and $t_2 = \Delta/3.0$ for the L_g phase. The P_n window was selected manually and generally ranged in length from 4 to 5 seconds, starting from about 1 second prior to the P_n arrival time. To get a smoother spectrum, the P_n window was extended to 20 seconds by zero-padding of the data. Noise spectra were calculated in a 30 second window preceding the P_n arrival. The signals were differentiated to acceleration and windowed using a 10% cosine taper between the limits defined above. The resulting acceleration spectra were divided by f to convert them to displacement spectra. If three-component data were available, the P_g and L_g spectra were each averaged using the vertical, radial and transverse components.

It is well known that spectral characteristics of seismic signals are strongly influenced by the background noise characteristics. In order to reduce the effect of noise on the signal spectra, only those frequencies for which the S/N level was greater than 2 were used. Frequencies at which the pre-event noise spectra exceeded the signal spectra by a factor of 2 were not used. Instead, these spectral points were obtained by fitting straight lines to the log spectra from the known neighboring points in the spectra.

The problem of missing data is a common problem in seismic analysis and sophisticated methods of dealing with missing data are important when discrimination is based with a vector of parameters (Glaser et al. 1986). However, in this study, since we used smooth spectral functions for inputs (rather than discrete measured parameters like m_b : M_s), a line fit to obtain missing points in the spectra was a simple solution that worked well.

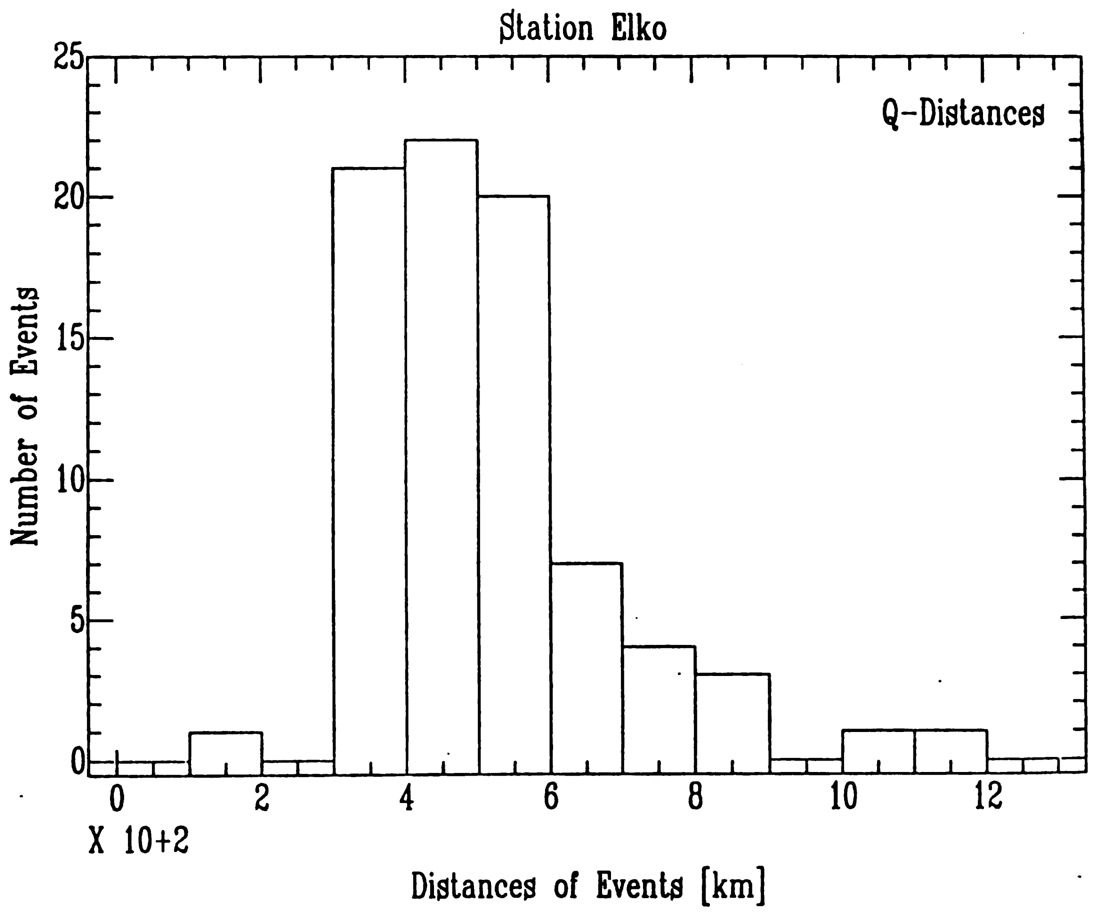


Figure 4.7: Histogram of the distances for the earthquake events from the Elko station. Note that the distances of the earthquakes varied from about 200 to 1200 km for this station.

4.5.2 CORRECTION OF DISTANCE EFFECTS ON THE SPECTRA

The distribution of the earthquakes in terms of distance is shown in Figure 4.7 for the Elko, NV recording station. The explosions were all from approximately the same site, about 400 km from the Elko station. Since lower frequency seismic waves propagate better through the earth, it is conceivable that these events could be discriminated solely on event distance rather than event type. Because the members of the explosion population set were approximately from the same location, and the earthquake data were not, we made a first order approximation to account for the effect of epicentral distance, *d*, on the source spectra. At frequency *f*, the observed spectrum P(*f*,*d*), is given by the relation:

$$P(f,d) = S(f)\exp\left[-\frac{\pi f d}{Q(f)\, v}\right] d^{-\kappa}$$

where S(*f*) is the source spectrum, $d^{-\kappa}$ for some constant κ is the frequency-independent geometrical spreading, *v* the group velocity, and Q(*f*) is the quality factor which is frequency dependent:

$$Q(f) = \alpha f^{\beta}$$

where α and β are constants which depend on the phase and on the regional geology. Since we assume that the $d^{-\kappa}$ term is frequency independent (and because we normalize the log spectra), a first order distance correction function for the spectra is simply

$$C(f,d) = e^{+\frac{\pi f d}{Q(f)\, v}}$$

Using results from studies applicable to the Basin and Range, we used $\alpha = 206$; $\beta = 0.60$ for P_g, and $\beta = 0.68$ for L_g; the velocity for P_g and L_g were 6.0 and 3.5 km/sec, respectively (Chavez and Priestly 1986). (We did not use any distance correction for the P_n phase and our final results are based on using only P_g, and L_g phases as input to the

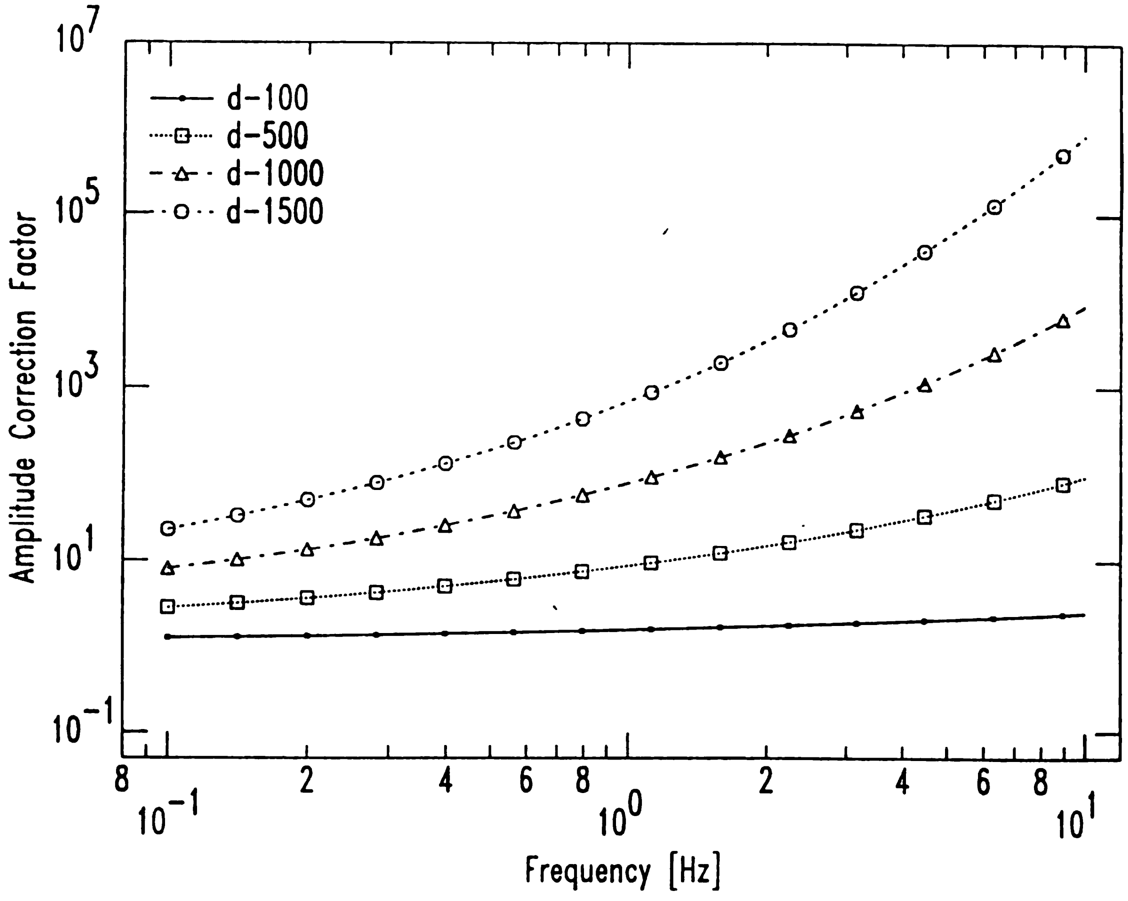


Figure 4.8: Spectral correction function for the L_g spectrum plotted as a function of frequencies and distances (D-km). Note that large corrections are made only for distant events.

network). A distance-corrected spectrum, obtained from the raw spectrum is given by

$$S(f) = P(f, d)\, C(f, d)$$

A plot of the distance correction function versus frequency and distance is shown in Figure 4.8. From Figure 4.8 we see that distance correction effects are stronger at higher frequencies and for distant events.

4.5.3 NORMALIZATION OF SPECTRAL LEVELS FOR MAGNITUDE INVARIANCE

As shown in Figure 4.9, on the average the explosions were of larger magnitudes than the earthquakes and certain pre-processing for reducing the effect of magnitude information in the data was appropriate. In order that discrimination is made on the basis of spectral shape and ratio, the input data to the network was formed by the distance-corrected log spectra of the three phases, P_n, P_g and L_g, normalized such that the maximum value of the input data was 1; i.e., the distance-corrected normalized input vector for the neural network was obtained by first applying distance correction to the spectra, then taking the logarithm, and finally normalizing the spectra such that its maximum value of the three spectral phases was 1. In summary, we wanted to be able to discriminate events by type, explosion or earthquake, and not by event magnitude; for a good test, the magnitude histograms of the two populations sets should be similar. However, since the magnitudes of the explosions were on the average larger than those of the earthquakes, we attempted to reduce the effect of event magnitude by normalizing the maximum value of each input pattern (the vector formed by the spectra of the three regional phases) to be unity.

Figures 10-a and 10-b show the statistical characteristics of the spectral data for the two populations. We draw attention to the fact, that for any single phase, there might be overlap. However, some

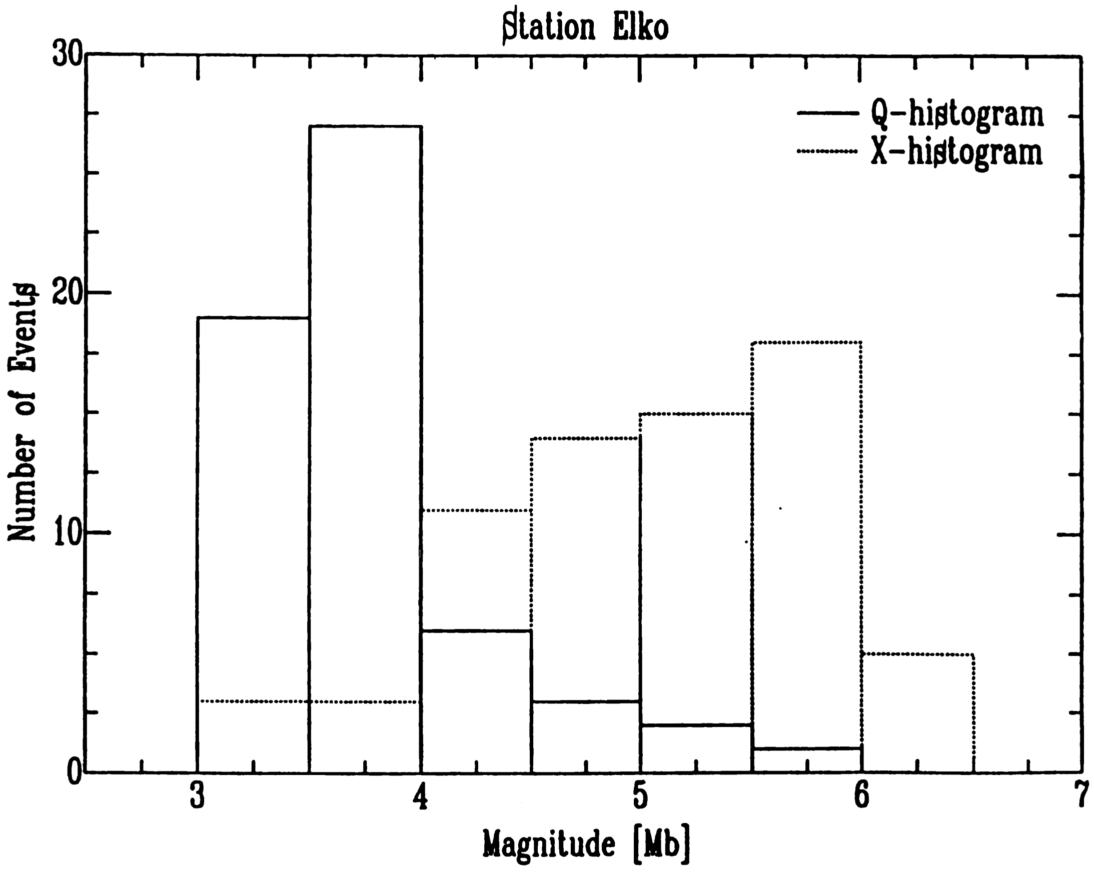


Figure 4.9: Distribution of the two populations (earthquakes and explosions) of the data base plotted as a function of the event magnitude in m_b for the Elko station.

distinct differences are observed between the two populations. For the explosions, the spectral shape of the L_g phase appears to have a higher corner frequency and a steeper decay than the earthquakes. Because the maximum amplitude was generally taken from the L_g phase, there is more overlap in L_g than for P_g. The normalized P_g spectra from the explosions are typically greater than the earthquakes which is consistent with a smaller L_g/P_g ratio for explosions over a broad range of frequencies, particularly between 1 and 5 Hz.

4.6 NEURAL NETWORK DISCRIMINATION PERFORMANCE

We experimented with multilayered neural networks architectures with a varying number of hidden units. We found that even with the simple network architecture (see Figure 4.3) consisting of an input layer, one hidden unit, and two output units, we could get very good performance. (Increasing the number of layers or hidden units did not increase system performance.) To regionalize the data characteristics, we assigned a separate network to each recording station.

4.6.1 TRAINING WITH CHEMOTAXIS AND BACK-PROPAGATION

Networks were trained both with the back-propagation and chemotaxis algorithms. Both training algorithms were able to find a decision boundary to separate all 170 events in the database at any of the four stations. Furthermore, they did so in a relatively short training time (approximately 500 - 2000 training cycles for either algorithm). Thus, computationally, there was no significant difference between the two methods.

In order to test the robustness of this decision boundary, we would like to test a trained network's performance on a test set of unclassified patterns. However, as we were working with a limited database of events (83 earthquakes and 87 explosions), it proved to be difficult to choose an appropriate subset of examples to use as a training set. Instead, we used a "leave-one-out" strategy (Lachenbruch and Mickey 1968) to determine the performance of the network. According to this strategy, the network was trained on all but one of the events in the database. Once trained, the network was tested on the event left out of the training set. To obtain a statistical measure of the discrimination performance, the process was repeated for every event in the database.

Performance evaluation via this method required training 680 networks (170 events times 4 recording stations). Thus, we needed to automate the process of generating training sets, training networks, and evaluating each network's performance. We used an "off-the-shelf" back-propagation simulator (SunNet) on a SUN 4/110 workstation for network training. Training took about four days on a dedicated computer. As we had established that the chemotaxis algorithm could train networks as well as back-propagation, we determined that there was no need to create an automated version of the chemotaxis algorithm solely to repeat these computations.

4.6.2 LEAVE-ONE-OUT PERFORMANCE

As shown in Table 4.1, preliminary results of this data set indicate that the rate of correct recognition for untrained data is from 93-97.5% for earthquakes and explosions at any single station of the four station network. When using a majority voting scheme with the network of four stations, the rate of correct recognition is over 97%. These results are based on using only the P_g and L_g spectra as input to the network. We found that inclusion of the P_n data did not significantly change the performance, and since the P_n had lower signal-to-noise ratio it was left out in the final analysis.

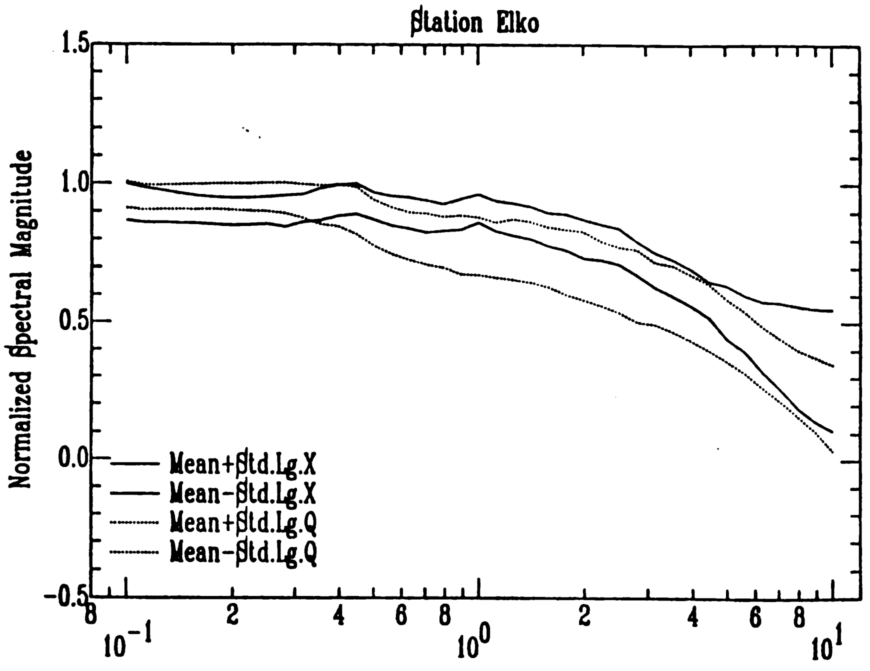
Out of the total 679 (earthquakes and explosions at 4 stations) spectra, the network either could not classify or mis-classified 29 spectra. The mean magnitude of these 29 spectra was 3.74 m_b with a standard deviation of 0.42. We can say, therefore, most of the

TABLE 1

Performance of the Artifical Neural Network

	Number of Events (Q/X)	Correct Identification (%)	Mis-Identification (%)	Undecided Classification (%)
Elko	80 / 79	97.5 / 97.5	0.00 / 2.50	2.50 / 0.00
Kanab	86 / 83	96.5 / 96.4	0.00 / 1.20	3.50 / 2.40
Landers	86 / 94	93.0 / 95.7	3.50 / 2.10	3.50 / 2.20
Mina	76 / 95	93.4 / 93.7	5.30 / 4.20	1.30 / 2.10
NETWORK	79 / 83	100 / 97.6	0.00 / 2.40	0.00 / 0.00

Table 4.1: Performance of the neural network for discrimination expressed in terms of percentage. In the table the entries Q/X represent results for earthquakes (Q) and explosions (X), respectively.



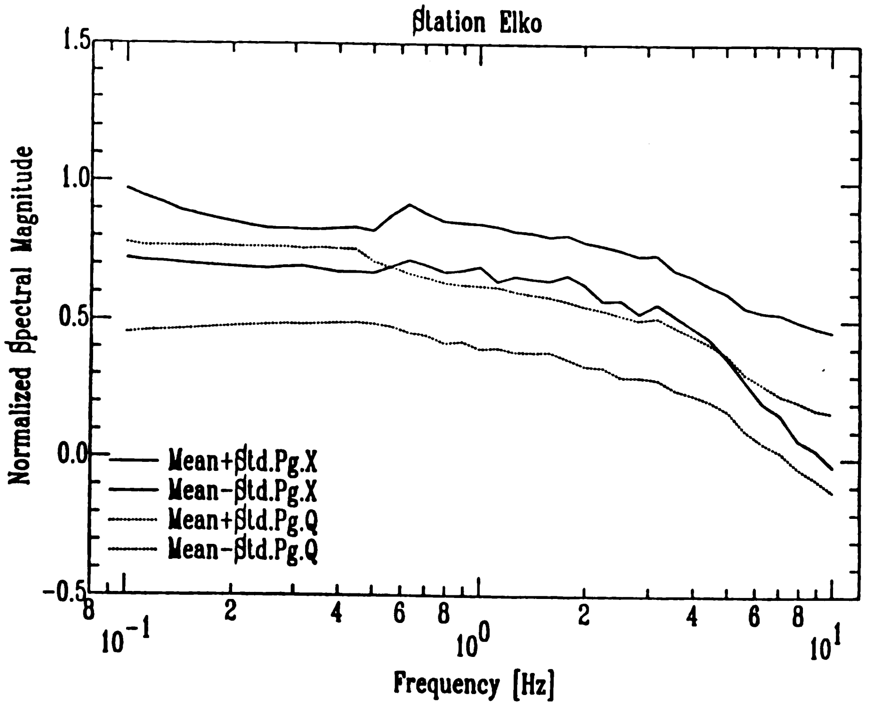


Figure 4.10 a-b: Plots of the normalized (a) L_g and (b) P_g spectra for events at the Elko station. Like lines are the *mean* ± *standard* deviation obtained from the complete data base for an event type. Solid-dashed lines denote the explosion spectra and dotted-dashed lines denote earthquake spectra.

events that were not correctly identified were among the small magnitude events.

4.6.3 DISCUSSION

While we are encouraged by these preliminary results, we note two defects in our present database. Figure 4.9 shows the histograms of the events in terms of their magnitude. Since we want to distinguish events by their type (earthquake or explosion) and not by the event magnitude, ideally, we want the histograms of the two populations to be similar. As the histogram shows, the magnitude of the explosions were on the average larger than those of the earthquakes. We attempted to reduce the effect of event magnitude by normalizing the maximum value of each pattern (the vector formed by the spectra of the three regional phases) to be unity. Because this normalization may not completely remove the effect of magnitude in the explosion or earthquake spectra, we cannot at present confirm whether the network was a magnitude-independent discriminator. The second peculiarity in our database was that the explosions were from the same site, while the earthquake locations had significant variations (see Figure 4.7). We tried to reduce the distance effect by a simple distance correction formula explained earlier. In any event, to test the validity of these preliminary results, we need to verify them against other databases. Finally, we note that appropriate pre-preprocessing (e.g., careful spectral analysis, distance correction, and normalization) was important for good discrimination performance.

4.7 COMPARISON WITH THE FISHER DISCRIMINANT

Since empirical results showed that a simple neural network architecture obtained results as good as those from networks with several hidden units, we might expect that this particular data set is almost linearly separable. This implied that conventional methods may be sufficient for the task. We applied the Fisher discrimination method (Tjøstheim 1981, Duda and Hart 1973) to the same data set for comparison of the results with conventional methods.

In order to use the Fisher discriminant, we must estimate a sample covariance matrix of the population and be able to invert the matrix. When the dimension of the input vector (≅ 80) is less than or equal to the number of events in each population set, it is numerically difficult to invert the sample covariance matrix. By assuming equal covariance distribution for explosion and earthquake populations, however, we were able to compute a better conditioned covariance matrix because of more sample events, approximately 160. Under equal covariance Gaussian distribution of the population sets we have

$$P(x \mid \text{explosion}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}}\exp\left[\frac{1}{2}(x-\mu_x)^T\Sigma^{-1}(x-\mu_x)\right]$$

$$P(x \mid \text{quake}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}}\exp\left[\frac{1}{2}(x-\mu_q)^T\Sigma^{-1}(x-\mu_q)\right]$$

where μ and μ_q are the sample mean vectors of the explosion and earthquake spectra, respectively, and Σ is the sample spectral covariance matrix. Assuming that the populations have an equal covariance Gaussian distribution as above, the log likelihood test for discrimination is given by

$$\text{Log}\left[\frac{P(x \mid \text{explosion})}{P(x \mid \text{quake})}\right] = x^T\Sigma^{-1}(\mu_x - \mu_q) - \frac{1}{2}(\mu_x + \mu_q)^T\Sigma^{-1}(\mu_x - \mu_q)$$

Results upon application of the Fisher discriminant are listed in Table 4.2. Because we used the criteria for a positive likelihood ratio event is an earthquake, there were no undecided events with the Fisher discriminant. (With a neural network, in contrast, high, medium or low responses in both output neurons meant that the classification was undecided. Because of these differences, a fair comparison over the network for the two methods becomes more complicated. as

TABLE 2

Performance of the Fisher Discriminant

	Number of Events (Q/X)	Correct Identification (%)	Mis-Identification (%)
Elko	80 / 79	96.3 / 96.2	3.7 0 / 3.80
Kanab	86 / 83	93.0 / 90.4	7.00 / 9.60
Landers	86 / 94	83.9 / 98.9	16.1 / 1.1
Mina	76 / 95	88.2 / 98.9	11.8 / 1.1

Table 4.2: Performance of the Fisher linear discriminant expressed in terms of percentage. As in table 4.1, the entries Q/X represent results for earthquakes (Q) and explosions (X), respectively.

such, for this preliminary study, we do not compare the network performance between the the two methods.) For any single station, the results on the same data set indicate that for the Fisher method rate of correct identification ranged from 88.2 to 98.9 per cent over the network for both earthquakes and explosions for any given station (see table 4.2).

The results of this simple multivariate Fisher discrimination method are nearly as good as those of the neural network. The comparable performance of the two methods is not surprising because with a very simple architecture for the neural network we obtained results as good as those from complex neural network architectures with many hidden units meaning that the two populations are almost linearly separable and therefore the performance of the Fisher technique should be quite close to optimal.

In summary, both neural network and Fisher methods perform quite well with regional spectral data, indicating perhaps that given complete spectral information, events with magnitude 4 or higher in this geological region can be discriminated rather well. Whether the same discrimination method will perform equally well in other geophysical environments or at lower m_b values is unknown.

4.8 CONCLUSIONS

In our study we deliberately kept the architecture of the neural network to the simplest form that would maintain performance. As it turns out, a single layer network was capable of realizing the same performance as a multilayer network. This allows for simple interpretation of the network weights as adaptive filters. Through examination of the network weights, we can find the features in the spectral data which the network utilized for discrimination of the two populations.

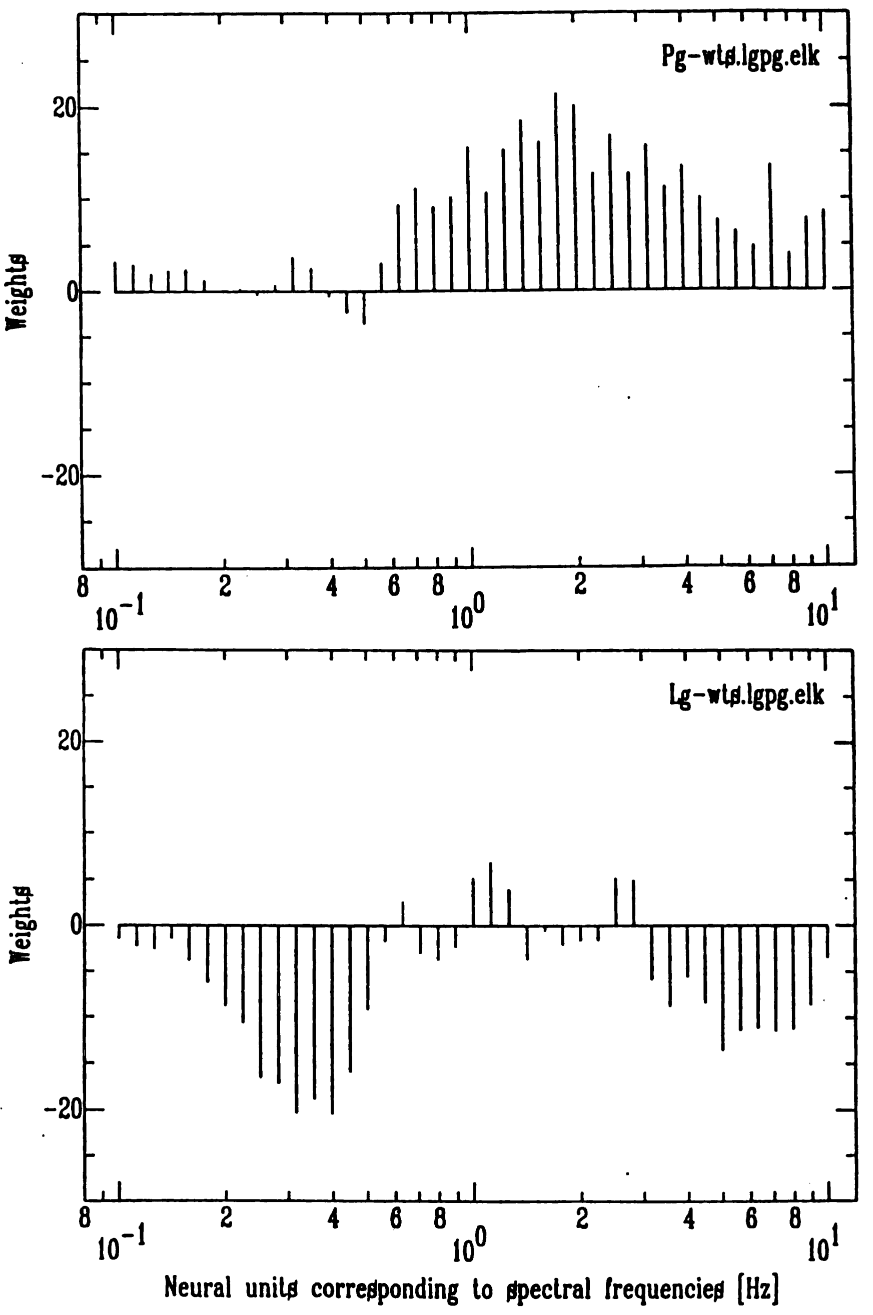


Figure 4.11: The weights from the input to a hidden unit for a trained network at Elko. Because the net had a simple architecture and the input units correspond to the frequencies in the spectrum, examination of the weights is useful for understanding how the network used features in the L_g and P_g spectra for discrimination. Since the log spectra of P_g has mostly positive weights and the log spectra of L_g has mostly negative weights, the network is using the L_g/P_g spectral ratio for discrimination, in addition to the spectral *shape*.

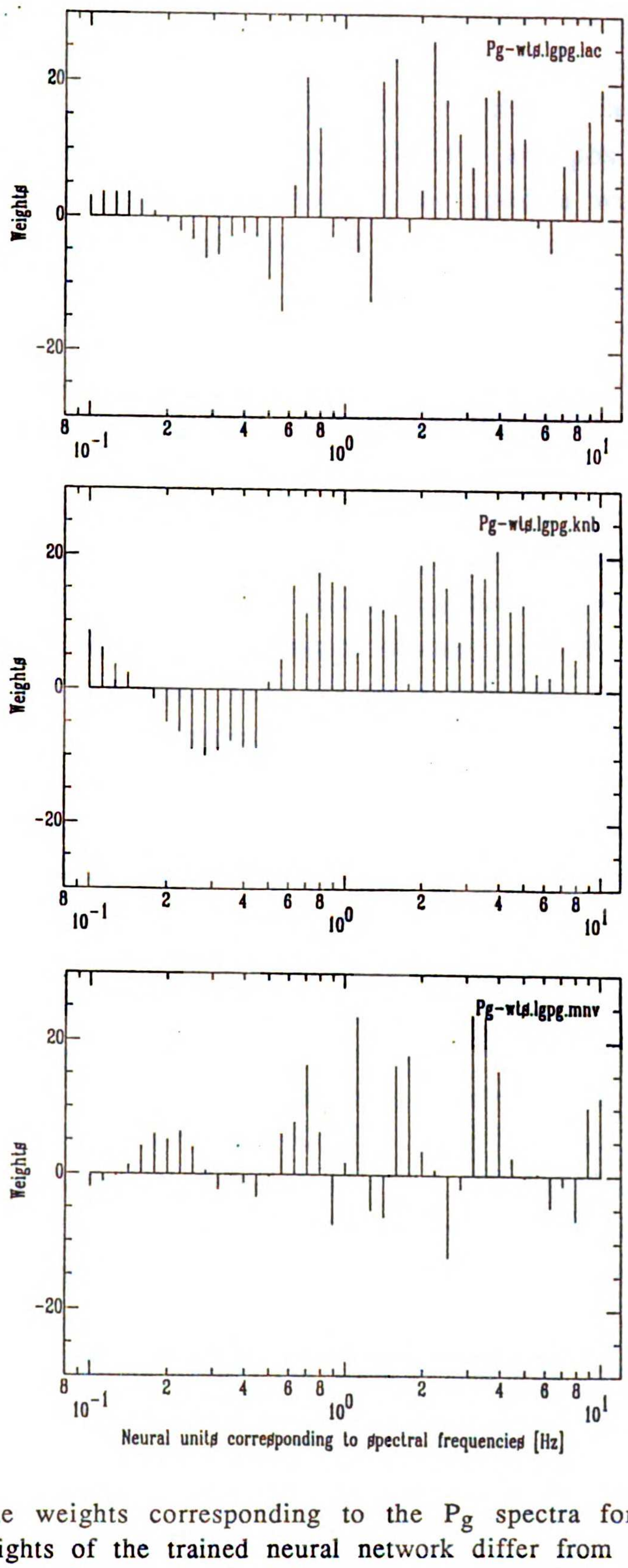


Figure 4.12: The weights corresponding to the P_g spectra for the other three stations. The weights of the trained neural network differ from station to station, supporting the fact that regional differences are important for determining optimal discriminant functions.

In Figure 4.11, we compare the L_g and P_g weights of the network for the Elko station corresponding to the L_g and P_g spectral inputs. Note that most of the weights for P_g are positive and those for L_g are negative. Since the log of the spectra was input to the network, this implies that the network has learned to distinguish an explosion from an earthquake by using the L_g/P_g spectral ratio. It is interesting, although not surprising, that the network through its learning process automatically developed a discriminant which had been previously recommended by experts in the field.

We investigated the variations of the input weights with the different training sets (using the leave-one-out approach the number of training sets is equal to the total number of earthquakes and explosions at a station.) The standard deviation of the weight vector about its mean showed that there is no significant variations in the weight distribution with different training sets of the leave-one-out method. Another interesting result was the fact that the weight vectors exhibited variations in the detailed shape between the four stations - although they had the same general shape. The P_g weight vectors for the other three stations are compared in Figure 4.12.

4.8.1 ADVANTAGES OVER CONVENTIONAL METHODS

Although the performance of the network for discrimination between explosions and earthquakes was quite similar to that of the Fisher linear discriminant, we found that the network exhibits a number of computational advantages over conventional methods. When the dimensions of the data are large (in this study the length of the input vector was 82), conventional methods can become numerically unstable and slow. For example, the Fisher technique requires the inversion of a covariance matrix, and it is well known that inversion of a large dimensional matrix can be numerically difficult if the matrix is poorly conditioned. Furthermore, from many other experiments that we performed we found that the network is quite robust to missing data points. Finally, a neural network provides a computational environment that is significantly more flexible and simple from the viewpoint of an application scientist.

4.9 SUBSEQUENT WORK ON SEISMIC SIGNAL DISCRIMINATION

There are many interesting questions and issues that we have not addressed in this preliminary study. Some of the problems now under study are: How important are the detailed spectral shape of the various phases? Was the signal-to-noise ratio (SNR) the main reason that the P_n data did not significantly contribute to the solution? How important is it to train separate networks station by station, or would it be better to use all the data from the seismic network and apply it to a single network? Work along these lines is continuing at the Earth Sciences Department at LLNL.

Dowla (Dowla, in preparation 1991) has since extended the neural network technique for the discrimination of *complete* seismograms or waveforms, instead of using detected, windowed phases. This makes the network performance more independent of an analyst's preprocessing and render the technique as an important tool in automatic seismic signal analysis. In his study, network input consisted of 700-800 points representing the short-term power spectra windowed at intervals over the entire signal. In this case, they found it necessary to use *multilayer* neural networks (with a minimum of 5-6 hidden units) to achieve the best performance. Because of the large network size and incomplete data, Dowla was only able to train on 50-60 events at each station, so that direct comparison with our study was impossible. Using the leave-one-out strategy, however, the new network was able to correctly classify 100% of the events in the training and test sets. (Remember that on any given network was trained on all but one of the events and tested on the other.) Dowla showed that multilayer neural networks yielded better results than linear regression techniques.

CHAPTER 5

RECURRENT NETWORKS AND THE CHEMOTAXIS ALGORITHM

5.1 INTRODUCTION

Thus far, we have limited application of the chemotaxis algorithm to feed-forward networks. The power of feed-forward networks as functional approximation devices has been amply demonstrated theoretically (Chapter 3) and in practical applications (e.g. Chapter 4). But these networks are only a subclass of the fully-connected finite-state automatons proposed by McCulloch and Pitts. As we noted in Chapter 1, if feedback loops are allowed in the network architecture, neural networks are potentially *universal* computational devices. These networks are referred to as "recurrent" networks, and are capable of a wide range of dynamical behavior, such as preservation of internal states, self-activation, lateral inhibition, refractory periods and time delays. (A typical recurrent architecture is illustrated in Figure 5.1.)

Many efforts are now underway to expand the realm of neural networks to intrinsically sequential or temporal signal processing problems. One approach currently being pursued is to *train* recurrent networks on sequential input/output patterns. Most recurrent network training algorithms are extensions of the back-propagation algorithm (discussed in Section 5.3). One advantage of the chemotaxis algorithm is that it does not require modification for recurrent networks other than in the definition of the objective function. We detail these modifications in Section 5.4. Next, we use computer simulations to demonstrate the effectiveness of the chemotaxis algorithm on three representative sequential tasks. We report the results of attempts to train recurrent networks to:

(1) integrate a sequential signal (Section 5.5.1),
(2) mimic a finite-state machine (Section 5.5.2), and
(3) oscillate at various frequencies (Section 5.5.3).

We note that all existing methods for training recurrent neural networks are extremely slow and not practical for many engineering applications. In the Discussion, we suggest alternative neural network strategies to deal with temporal problems.

5.2 RECURRENT NETWORK THEORY

A recurrent network is equivalent to a finite-state automaton (Arbib 1987, p. 25). Much is known about what these structures can do *in principle*. McCulloch and Pitts (1943) asserted that networks connected "with circles" (feedback loops) could compute any numbers computable by a Turing machine even *without* added scanners or an infinite tape (see also Chapter 1; Turing 1937; Arbib 1964,1987; Cowan 1990). They did not, however, provide a proof of this assertion.

At first, one might think that construction of a universal Turing machine out of fully connected neurons could only be possible provided a potentially infinite number of neurons are available. But one can encode an infinite tape into a finite tape, provided that that tape has real valued activations. Indeed, Seigelman and Sontag (1991) showed that it is possible to build a recurrent net that simulates a universal Turing machine with a *finite* network of *sigmoidal* neurons. (They estimate the network would consist of less than 10^5 neurons.) This is an important finding, because it shows that, *potentially*, recurrent networks can simulate an arbitrary computing device (i.e. recurrent networks are capable of computing anything computable by a sequential computer).

As is the case for the theoretical results for static neural networks, most of these results are existence proofs (not known for their usefulness in practical applications). In contrast to feed-forward neural networks, however, efficient training algorithms for finding these solutions are not yet available. For example, recurrent neural networks have been trained to mimic functions of several finite-state machines (Elman 1988; Williams and Zipser 1988; Section 5.4.2). However, it is much easier simply to *design* finite-state machines than to train what is essentially a randomly connected finite-state automaton to achieve the same input/output relationships. This is demonstrated in section 5.5.2.

On a less general level, we would like to know what recurrent neural networks could do *practically*, in other words, "what are they best suited for?" A more recent theoretical result claims that recurrent neural networks are universal *controllers*[1]; that is, for any controllable system (as defined in linear control theory), there exists a recurrent neural network implementation of the corresponding ideal controller. In this area, recurrent networks are yielding some initial successes (Seidl and Lorenz 1991, Swiniarski 1991).

5.3 TRAINING ALGORITHMS FOR RECURRENT NETWORKS

Recent efforts to train recurrent networks to generate or classify temporal patterns (for example adding two sequential inputs [Tsung and Cottrell 1989] or memorizing oscillatory patterns [Doya and Yoshizana 1989]) have relied on extensions of the back-propagation learning algorithm (Almeida 1987; Pineda 1988; Williams and Zipser 1988; Elman 1988). (For a comprehensive review, see [Williams and Zipser 1988]).

One of the early schemes relied on *back-propagation-through-time* which limited the duration of the signal to the number of

[1]Robert Hecht-Nielsen announced that there is a paper (not his) currently under review for publication that proves this result (at the AMSE Int'l Conf. Neural Networks, San Diego, May 29-31, 1991).

processing layers (Rumelhart and McClelland 1986). Other learning algorithms require specific architectures (Elman 1988, Jordan 1987). The learning rules presented by Almeida (1987) and Pineda (1988) are limited in that they require that desired network dynamics have only point attractors. A more general learning algorithm, recurrent back-propagation (Williams and Zipser 1988), places no constraints on how the network is connected or the dynamics of the neural signals.

Recurrent back-propagation requires that the objective function E(w) be redefined to account for output error as *a function of time* E(w,t). So for each weight w_{ij} in the network, a value

$$\Delta w_{ij}(t) = -\alpha \frac{\partial E(\mathbf{w},t)}{\partial w_{ij}}$$

is accumulated for each time step Δt along the output trajectory (α represents the algorithm learning rate). After the network has run through its trajectory, each weight is altered by

$$\sum_{t_0}^{t_f} \Delta w_{ij}(t).$$

In summary, recurrent back-propagation requires that gradients be calculated and summed over the duration of the simulation.

One variation of the recurrent back-propagation algorithm, called 'teacher-forcing', seems to be helpful in learning some dynamical behavior, such as learning stable oscillations (Williams and Zipser 1988). In 'teacher-forcing' during training, the actual output of the network is replaced by the teacher signal creating a 'teacher forced state'.

Back-propagation, already considered biologically implausible for *static* neural networks, has yet to be extended to dynamical networks in a biologically realistic way. Recurrent back-propagation is even less plausible biologically, since it requires credit assignment backward through *time* as well as through the network. One advantage of the chemotaxis algorithm is that it does not require modification for recurrent networks other than in the definition of the objective function. In this respect, *it is a general learning rule.* Thus initially, we had great enthusiasm about extending the chemotaxis algorithm to recurrent networks.

5.4 MODIFICATIONS OF THE CHEMOTAXIS ALGORITHM FOR RECURRENT NETWORKS

5.4.1 NETWORK ARCHITECTURE

In it's most general form, a recurrent network allows all elements to connect with all others, including themselves. (Figure 5.1 shows the network architecture used in this study.) As in the case of static neural networks, net activation of a unit is a weighted sum of all of its inputs:

$$\mathrm{net}_i(t + \Delta t) = \sum_{j}^{J} w_{ij} o_j(t) + \sum_{k}^{K} w_{ik} I_k(t)$$

where $o_j(t)$ = the output of unit j

$I_k(t)$ = input signal k

The output of a unit is function of its activation on the previous time step. In this study, this *activation function* is a nonlinear "squashing function" of the form:

$$o_i(t + \Delta t) = \left(e^{-\mathrm{net}_i(t)} + 1)\right) \Big/ \left(e^{-\mathrm{net}_i(t)} + 1)\right)$$

or

$$o_i(t + \Delta t) = 1 \Big/ \left(e^{-\mathrm{net}_i(t)} + 1)\right)$$

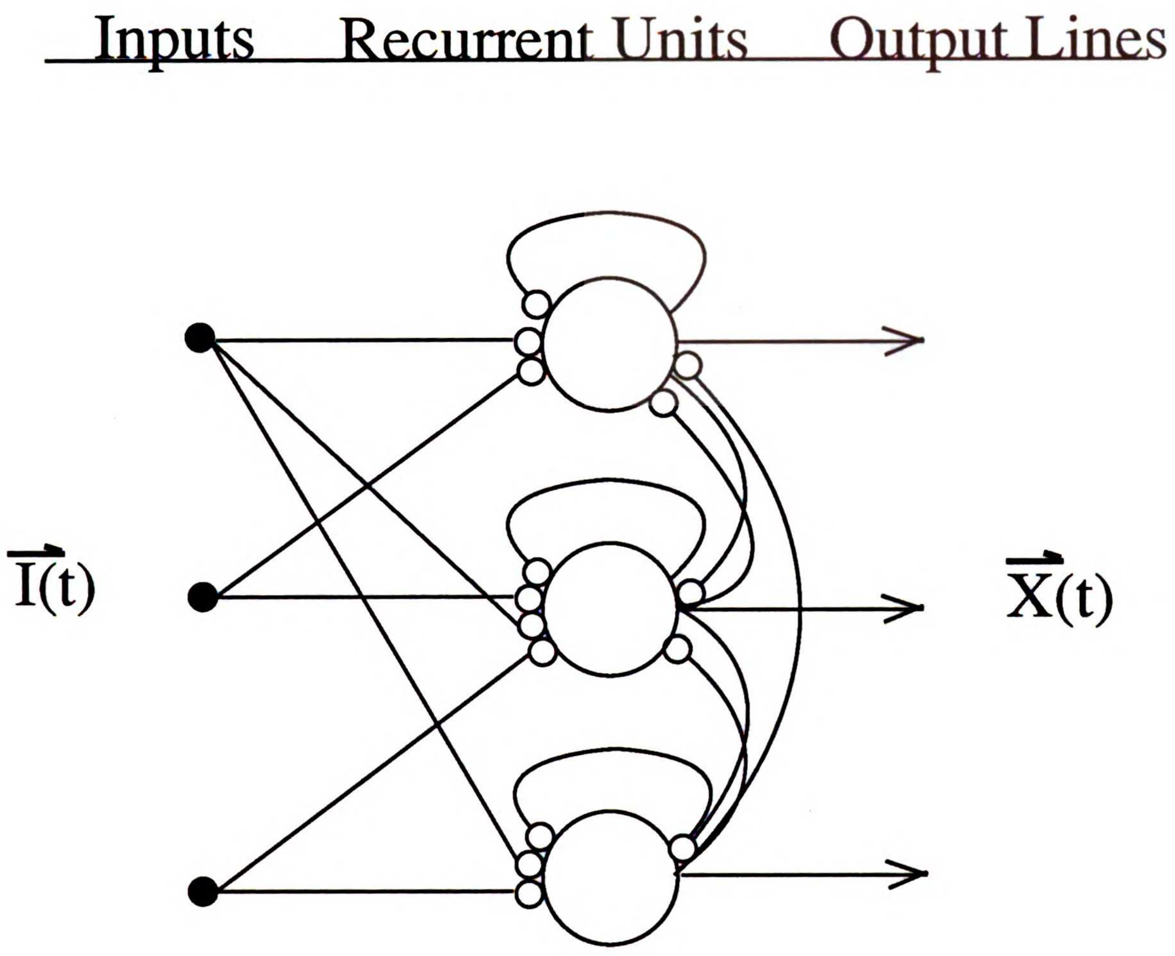


Figure 5.1: Architecture of a recurrent neural network

All units are allowed to receive input from any other unit - including themselves. External inputs can be arbitrarily assigned. Training requires the evaluation of the performance of at least one unit's response.

Note that the networks studied here are given non-linear, continuously differentiable activation functions; however, the algorithm does not require that this be the case. Back-propagation and its variants use local gradients to *predict* favorable directions in weight space. Since the chemotaxis algorithm chooses random directions in weight space, it does not require differentiable activation functions.

5.4.2 DEFINITION OF THE OBJECTIVE FUNCTION

Notice that a temporal dimension has been added to the input and output patterns. The simplest way to incorporate this into the training algorithm is simply to integrate, or sum, the error between the target signal dynamics and the output trajectories. Hence, we defined the total network error as follows:

$$E(w) = \sum_{p}^{P} \sum_{t=0}^{t_{max}} \sum_{n}^{N} \{x_{np}(t) - T_{np}(t)\}^2$$

where:	P	=	the number of patterns in the training set
	t_{max}	=	duration of the target sequence
	N	=	number of targeted output units
	T_{np}	=	target response of unit p to pattern n
	x_{np}	=	actual response of unit p to pattern n

We note that there is considerable flexibility in choosing the exact form of the error function by which the network performance is evaluated. The output signals can be sampled, for example, at every tenth time step or asynchronously. Furthermore, the error could be weighted toward the final state reducing the importance of the actual trajectory taken to reach the desired state (i.e. ignoring the transient response.) In the following simulations, the output error is unweighted and calculated at each time step.

5.5 SIMULATION RESULTS

5.5.1 A NEURAL 'INTEGRATOR'

For an initial test of the training algorithm, a small network was trained to integrate an input signal I(t). Notice that it is trivial to *design* a linear network to perform this task. (A linear unit with a feedback weight of "1" approaches an ideal integrator as $\Delta t \rightarrow 0$.) The purpose of this demonstration was to verify the training procedure.

The network was trained on 8 representative patterns whose magnitudes ranged between 0 and 1 over ten discrete time steps (see Figure 5.2a). Although the maximum output range of the net was -1 to +1, no negative valued input signals were used in the training set. The target signal for the output was chosen to be

$$T(t_i + \Delta t) = K \int_0^{t_{max}} I(t_i)\, d\tau$$

with K, a scaling constant, set at 0.095 to keep the output within the dynamic range of the output unit. Since it takes at least one time step for information to propagate from the input node to the output, network error was calculated with a delay of one time step.

The net architecture was '1-H-1' (one input node representing the input signal, H fully-connected hidden units, and one output unit at which the network response was measured).

Figure 5.2a shows the performance of a trained 1-4-1 net after 500 cycles through the training set (epochs). Training time was independent of the number of hidden units. Training time was also independent of the *length* of the input signal (number of sample points). The total squared error of the net on this training set is 0.2. Notice that the net performance eventually saturates when it tries to follow a maximal step input (Fig. 2a-i). Here the output unit's

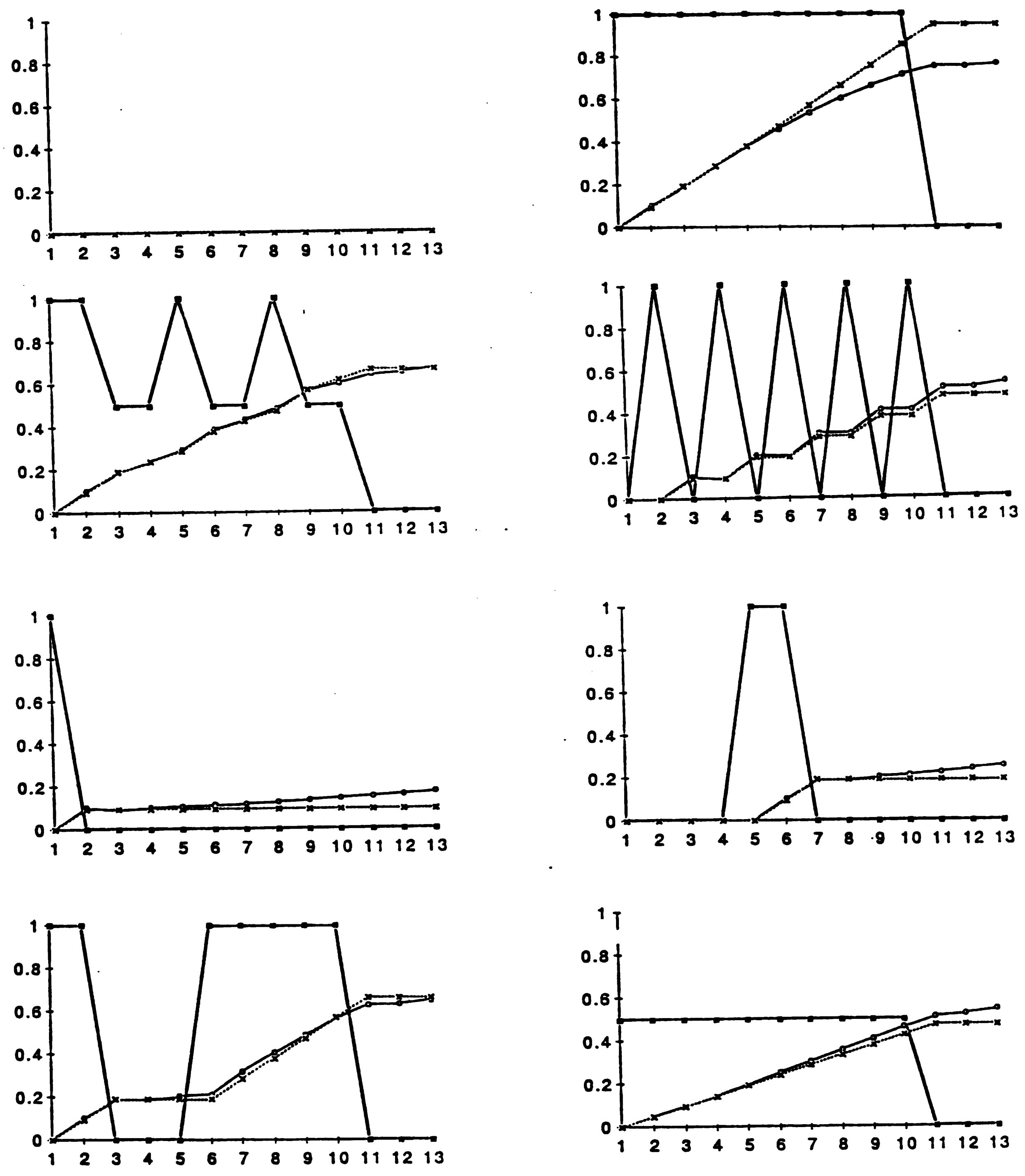


Figure 5.2a.: Performance of the neural integrator on the training set.

Eight temporal signals were used to train a network to integrate an incoming signal. The heavy solid line indicates the input signal and the crossed-dashed line, the time integral of the input. The light, solid line shows the corresponding output of a trained network.

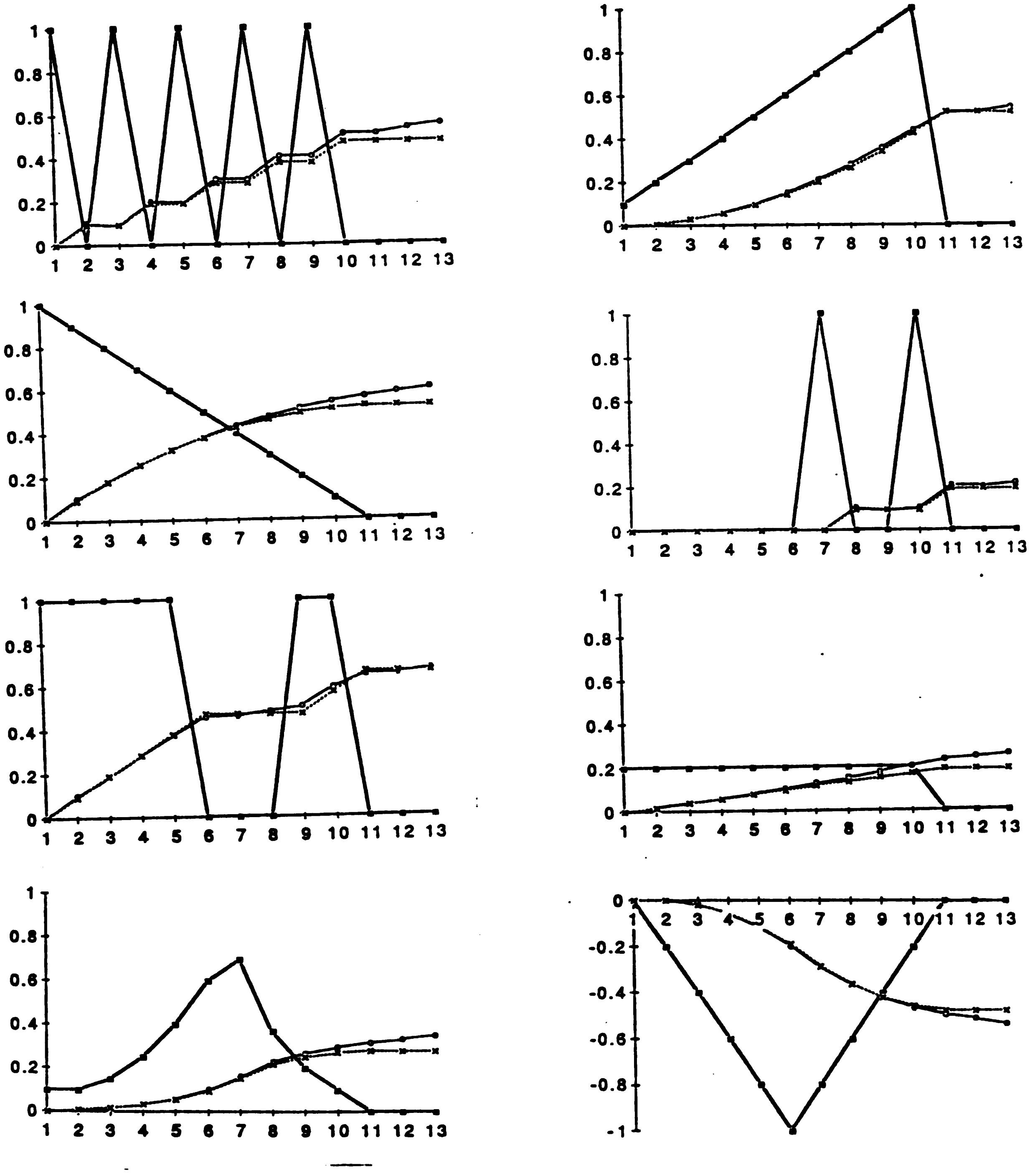


Figure 5.2b.: Performance of the 'neural integrator' on the test set.
Note that the trained network responds correctly to negative valued inputs which were not included in the training set.

response is being limited by the nonlinearity of its activation function.

Figure 5.2b shows the response of the net to various test signals. Notice that the net responds correctly to a ramp input, a high frequency input as well as *nonpositive* inputs.

5.5.2 TRAINING A RECURRENT NETWORK TO BE A FINITE-STATE MACHINE

A recurrent network with a linear threshold activation function can be considered equivalent to a finite state machine. In this study, we explore the practicality of *training* a recurrent network to achieve a realization of a particular finite-state machine - a digital 'flip-flop'.

The state transition matrixes for a flip-flop is shown in Figure 5.3. Notice that the required operations are analogous to the 'exclusive OR" (XOR). In other words, the two-state machine will perform the task of evaluating the parity value of a sequential, digital input signal. The trained network must learn to generalize a task for which there are an *infinite* number of examples of input/output patterns. Also, as the number of recurrent units increases, the network will be capable of more internal states making the occurrence of 'spurious states' or minima more likely. Thus, the problem is to train the smallest network possible on a small subset of the possible binary sequences. Since there are only two internal states, there are only 4 combinations of transitions the network needs to learn. This suggests that the minimal training set size would consist of 4 patterns - one for each state transition case. We also note that as few as two units are sufficient to build a two-state machine. The following experiments were conducted to determine the practicality of creating a given FSM through training a recurrent network on examples of the desired dynamical input/output behavior.

Figure 5.3: State Transition Matrices for a Two-State FSM

$$\text{Next State} \qquad\qquad\qquad\qquad \text{Output}$$

$$\text{Input}\;\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 0 \end{array} \qquad\qquad \text{Input}\;\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 0 \end{array}$$

The networks in this study consisted of 2 to 4 fully connected units. One unit was arbitrarily chosen as the output unit. The single input to the network was a string of 1's and 0's. The target output was set to respond with a "1" if the total number of 1's received (after a clock-cycle delay) was even, "0" otherwise. Training continued until all of the output sequences were within 0.1 of the target sequence at each time step for every pattern in the training set. The network was then tested on an expanded set of patterns. The training sets were expanded until the trained networks had properly "generalized" the problem (see table 4.1).

First, a 2-unit network was trained on the minimum number of examples, that is the 4 patterns in the truth table for "exclusive OR" in sequential form. Although it was possible to train a network to do this task, the resultant networks would give inappropriate responses when tested on the same sequences shifted by only one time step. It seems that there was a strong tendency for the network to simply generate the 4 output sequences exploiting some of the initial conditions provided by the internal biases as well as the input patterns. To steer the network performance further toward the stated goal, the training set size was expanded to include both the 4, 2-bit patterns of sequential XOR as well as the same patterns shifted in time one time step. Thus, the network was trained on all 8 permutations of a three-bit signal (or 3-bit parity). Attempts to train a 2 unit network were unsuccessful. Training was possible, however, for larger networks. In addition, these networks performed well on test sequences. (The networks tended to give responses closer to the 'right' answer - within 0.5 - than the 'wrong' answer.) In addition, the networks could be forced to generalize to longer input sequences by continuing training for a few hundred cycles through the larger set. Networks trained on an even larger training set (4-bit parity) generalized well - often performing as well on 'unknown' sequences as on the training set. Figure 5.4 shows the final architecture of one such training run.

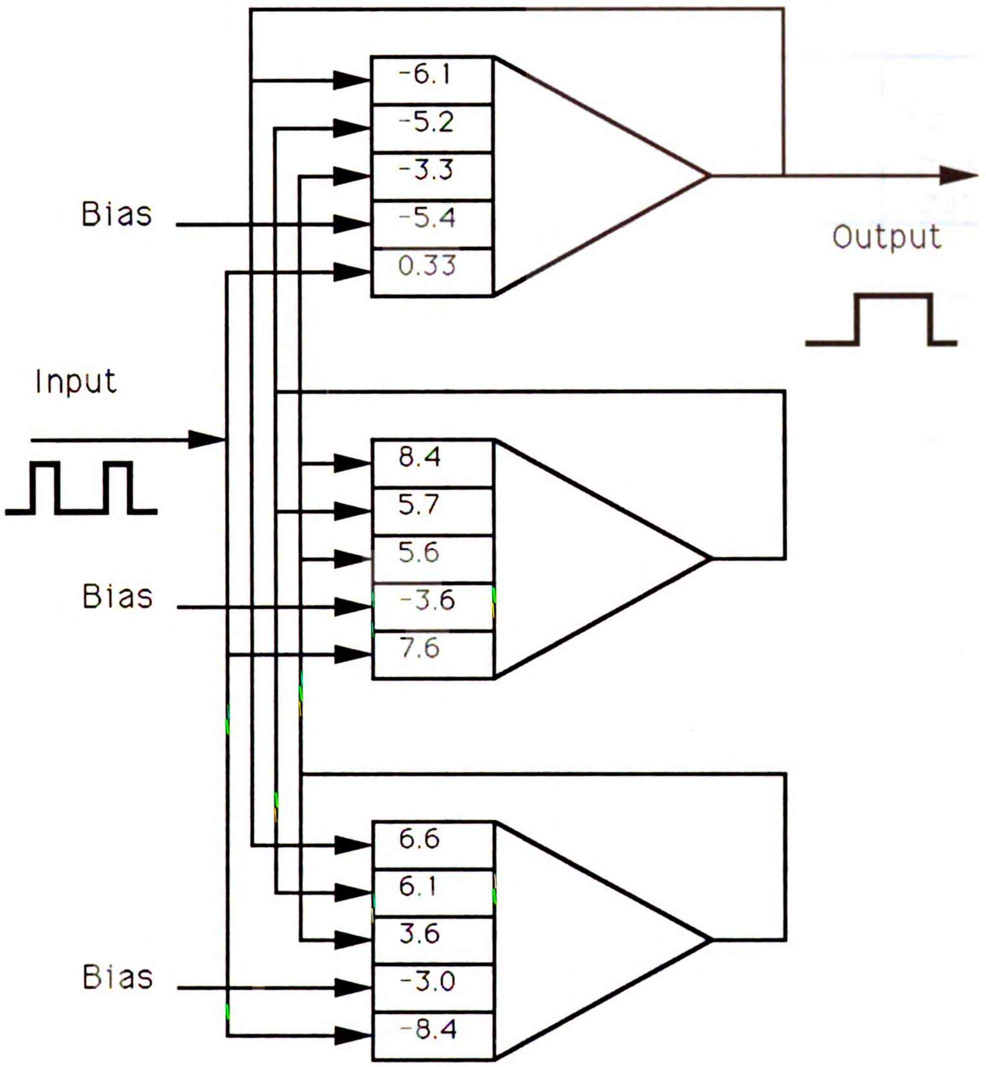


Figure 5.4: Training a recurrent neural network to be a finite state machine

A 'flip-flop' can be considered a 2-state finite state machine (FSM) which performs a parity operation on a sequential signal. The corresponding state transition matricies are shown in Table 5.1. An implementation realized by training a recurrent neural network is shown below.

Table 5.1: Average Training Times for Sequential Parity (1)

Training set Size	# of units	Average Training Time	Ratio (Successes to trials)	FSM Generalization ?
4 (XOR)	2	36,790	1/5	no
	3	****	--	--
	4	1219	5/5	no
8 (3-bit parity)	2	****	0/5	--
	3	3406	1/5	almost
	4	2132	8/10	almost
16 (4-bit parity)	2	****	0/5	--
	3	1859	2/10	yes
	4	3728	6/10	yes

(1)Averages were taken over successful training runs under various conditions (number of units and training set size). The ratio of successful runs to trials is given in the adjacent column. Training was stopped when all of the output sequences were within 0.1 of the target sequence at each time step for every pattern in the training set.

Training on the smallest possible training set (the 4 patterns in the XOR truth table) could be achieved fairly quickly; however, the resultant network would give inappropriate responses to the same sequences shifted in time. A network was judged to have "almost" generalized when it's response to test inputs are close to the desired output (within 0.5 instead of within 0.1). Such networks could usually be made to generalize by training the sub-optimal network on an expanded training set (≈200 cylces). A network was considered to have properly generalized the task of the 2-state FSM if its performance on test sequences was equal to that on the training set.

Only mathematical analysis of the final network architecture can guarantee that the network had properly 'generalized' the task of sequential parity from sample input/output pattern matchings. We note that a finite state machine is relatively trivial to *design*. We suggest that the power of recurrent networks, if it exists, may be in training dynamical systems to desired specifications and tolerances. This theme is pursued in the following sections.

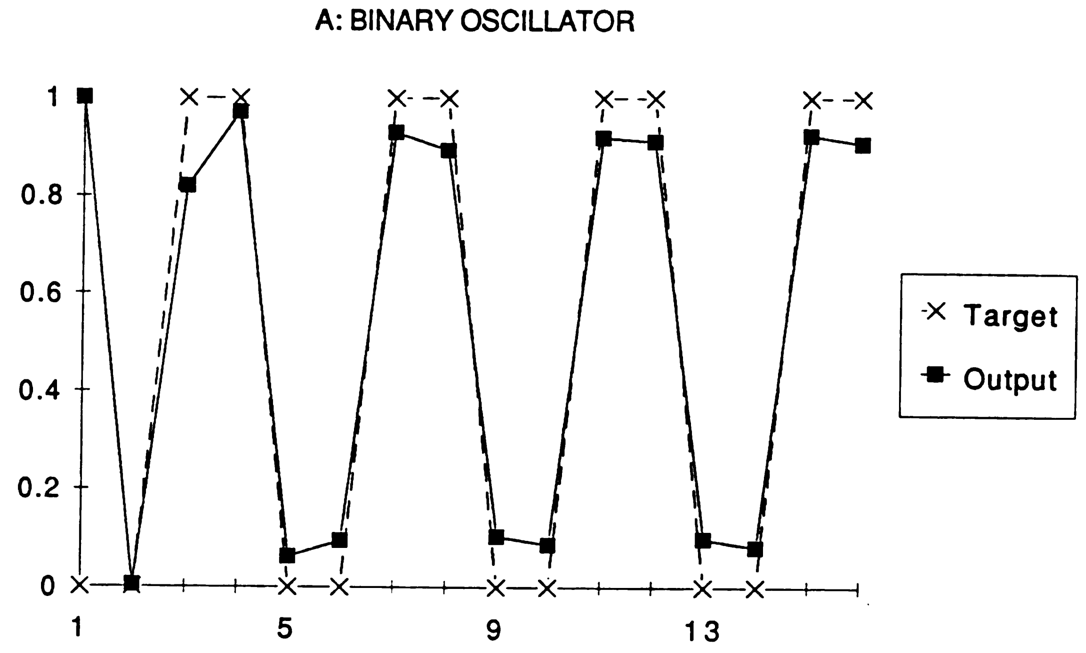
5.5.3 OSCILLATORS

Networks were trained to generate oscillations of specified frequencies and characteristics. The networks in these studies had no inputs and only one output.

In the first task (from Williams and Zipser 1988), a pair of logistic units were trained to produce the sequence 0, 0, 1, 1, 0, 0, 1, 1, ... (Period = 4 clock cycles). Training continued until the output was within 0.1 of the desired sequence. With initial random weights drawn from a uniform distribution [-1,+1] and biased units, the network learned this behavior in 2000 training cycles. Figure 5.5a shows the trained network output.

For comparison, Williams and Zipser (1988) report that simple recurrent back-propagation failed to converge at all on this problem. However, using a more complicated variation of the recurrent back-propagation algorithm, called 'teacher forcing' (Section 5.3), they were able to train a two unit network in around 100 training cycles.

In the second task, a 2 unit network was trained to generate a continuously valued sine wave oscillation with a period of 25 iterations. The training signal consisted of one complete cycle. The units in this study again had a logistic activation function, but this time the allowed range of operation was [-1,+1]. The network was trained until all 25 points were within 0.2 of the target sequence. (Note that it would be impossible for this system to generate a



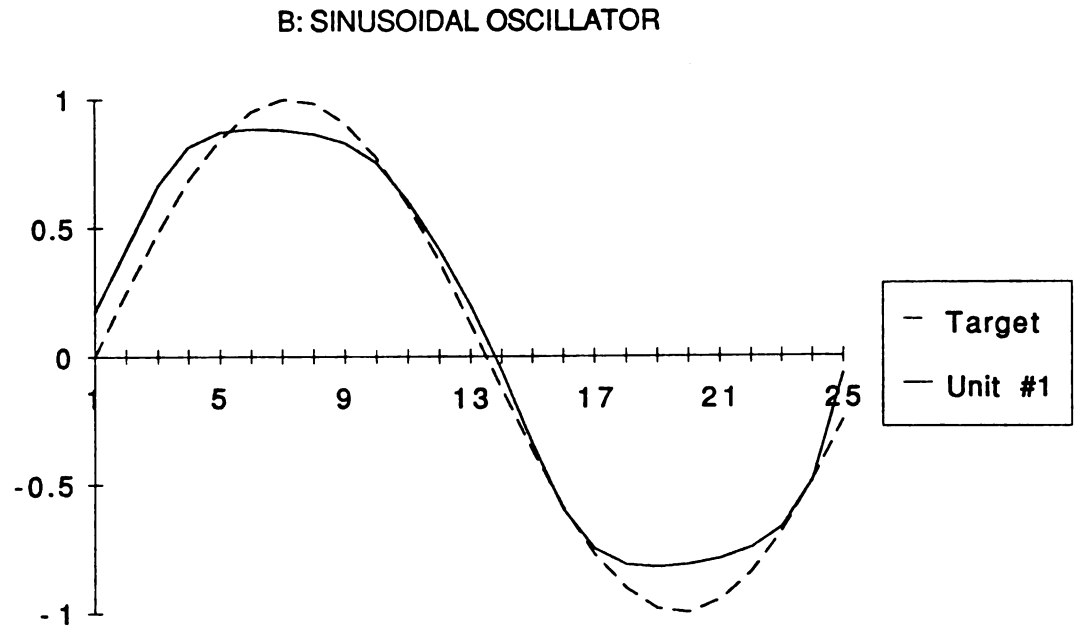


Figure 5.5 A & B: Performance of neural networks trained to oscillate

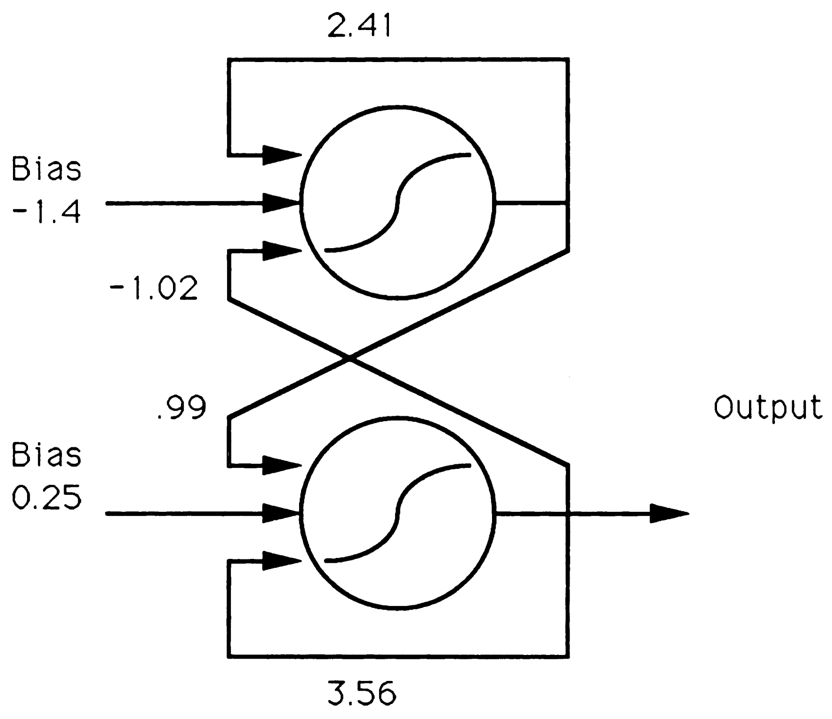


Figure 5.6: Implementation of a roughly sinusoidal oscillator with nonlinear, recurrent units (see also figure 5.5B).

perfect sine wave because of saturation effects.) Figure 5.5b shows the output of this network after 26,500 training cycles.

Training for this task proved to be surprisingly difficult. First of all, initially the training set consisted of a sample of two or more cycles. Training often failed to converge because of what might be termed 'phase washout'. If the network produced an output with a sightly different frequency than the target signal, the error over several periods will cancel out. Also, there seems to be a strong attractor part way through training where the network produces a damped response followed by a constant valued output. Williams and Zipser report similar results in their studies.

5.6 DISCUSSION

Recurrent networks are *in theory* capable of a wide range of dynamical behavior. These systems have potentially infinitely many states or attractors. Training recurrent networks requires a search of state space for the required attractor. We provided one example with our attempt to train a network to be a two-state finite-state machine. (A network consisting of 4 units was capable of occupying approximately $2^4 = 16$ binary internal states.) Thus, successful training required the use of long input/output sequences in the training set to keep the network from developing "spurious states". Hence, great care must be taken in the design of a recurrent network application for it is likely to succeed even at the most trivial applications. Training a *tabula rasa*, randomly connected, finite-state automaton to perform a *complex* sequential signal processing task may be a hopeless endeavor.

It is for these reasons, and not because of the inefficiency of the training algorithms, that recurrent network architectures must be constrained to "custom-fit" the application at hand. The most successful applications have applied recurrent networks by restricting their function in a control or filtering process.

For example, recurrent network structures have been proposed by which each output unit estimates a state variable $x_i(t)$ of a dynamical system (Seidl and Lorenz 1991; Swiniarski 1991). The structure used by Seidel and Lorenz consists of an internal formation of a two layer, feed-forward neural network which is used to approximate the next state as a function of the previous state vector $x_i(t + \Delta t) = f(\mathbf{x}(t))$. Using this less general architecture, they are able to prove that this approximation can become arbitrarily close to a specific class of nonlinear dynamical systems. Swiniarski used the same two-layer scheme to employ recurrent networks as "neural Kalman filters", where the network is trained to estimate the unobservable system states (Swiniarski and Nieplocha 1991).

5.6.1 ALTERNATIVES TO RECURRENT NETWORKS IN CONTROL

Many studies have employed feed-forward neural networks in feedback control systems (A wide range of applications appears in the April 1990 issue of *IEEE Control Systems Magazine.*) A common application is to use the neural network to estimate a nonlinear transfer function of either the feedback or feedforward controller (F. Chen 1990, Psaltis et al. 1988). Training the network increases the control system performance; thus this technique can be considered a nonlinear adaptive control process.

For many control problems, however, what we would like is a dynamical system which generates a temporal control signal as a function of modulating inputs. The existing recurrent training algorithms have not been shown capable of producing such a rich response.

We propose an alternative strategy to recurrent networks for motor control which employs static, feed-forward networks. We propose to exploit the function approximation capabilities of feed-forward networks (Chapter 3) to estimate the *parameters* of a dynamical control signal. One can think of this strategy as analogous to generating a temporal sequence by estimating the magnitudes of it's Fourier spectral coefficients. In the next chapter, we present methodology by which a *static*, feed-forward neural network can be used to control a *dynamical* system.

This control scheme has a biological correlate in high level motor control. It is well known that ganglia of invertebrates and spinal cord of vertebrates are capable of generating autonomous movements (such as walking in cats and coordinated movement of the lobster stomatogastric ganglion). These neural structures are called central pattern generators (For a review, see Selverston [1980]). Efferent fibers from higher central nervous system centers seem to only modulate this action by speeding up, inhibiting or reversing the basic motor patterns.

This neural network strategy assumes an ensemble of existing dynamical central pattern generators. In this context, the static neural network represents the higher level control, or modulation, of this ensemble. Thus, learning is the process of coordinating the interactions of existing function generators. This is somewhat akin to Edelman's selectionist hypothesis of neuronal "groups" in that learning entails the selection or "education" of an existing repertoire of dynamical groups (Edelman 1987, Crick 1989).

CHAPTER 6

NEURAL NETWORKS APPLIED TO OPEN-LOOP, DYNAMIC CONTROL[1]

6.1 INTRODUCTION

The purpose of this chapter is to explore the usefulness of neural nets in generating temporal signals for controlling dynamical systems. This is a universal problem in engineering (from controlling a chemical process to landing on the moon) as well as in biology (from regulating blood sugar to throwing a baseball). Conceivably, one could employ recurrent networks to generate control signals, since they are capable *in theory* of a wide range of dynamical behavior. But as we noted in Chapter 5, training methods for recurrent networks are extremely slow and inefficient for large dimensional problems. Here, we develop an alternative methodology for the generation of temporal control signals.

The specific control tasks considered here are open-loop, time-optimal control. We have chosen this problem because the optimal control signals can be derived from classical control theory, and thus we can assess performance against the ideal control. The usefulness of neural networks, however, will be in cases where the dynamics are nonlinear or may not even be explicitly known. Therefore, this methodology may be applicable to a wide range of control problems.

First, we reduce the dimensionality of the control problem by parameterization of the control signals. Next, feed-forward networks are then trained to approximate these parameters as a function of

[1]Some of the results of this chapter are to appear in The International Journal of Neural Networks: Research and Applications (1991).

the initial and final system states. Finally, we train networks to drive an unknown plant using error in the dynamical state trajectory as a performance measure. We note that a whole class of control problems, linear and non-linear, can effectively be solved by this method.

6.1.1 PARAMETERIZATION OF TEMPORAL SEQUENCES IN NEURAL NETWORK APPLICATIONS

In most successful applications of neural networks to temporal signal processing, the time signals are first transformed into *static patterns* by sampling in a time window. The sampled values are then used as inputs to the networks. This method has been applied to solve non-trivial problems such as pronouncing phonemes from strings of letters from ordinary English text (Sejnowski and Rosenberg 1988) and recognizing spoken words from sonograms (Lippmann 1989, Waibel 1989).

Conversely, the technique of transforming temporal signals into a parameter space (such as the frequency domain) can be employed for the *generation* of temporal signals.

6.1.2 PARAMETERIZATION OF TIME-OPTIMAL CONTROL SIGNALS

We wish to find an appropriate parameterization for a class of optimal control signals. Conceivably, this could be done using the Fourier spectral coefficients of the temporal control signal. Since time-optimal control signals often consist of discontinuous pulses or step functions, several spectral coefficients may be required for a good *approximation* of the control signal. In this section, we propose a parameterization which gives an *exact* fit with relatively few variables.

For concreteness, we consider the problem of controlling a simple ballistic (open-loop) movement. In the next section (6.2), we review optimal control theory and show how we can use Pontryagin's maximum principle to find the form of the optimal control signal. Application of the maximum principle gives us the well-known facts that (a) the time-optimal control of this system is *bang-bang* (the control variables only operate at their maximal allowed values) and (b) under certain conditions, the control variables have a finite number of changes between these extrema over the optimal trajectory. The changes in extrema occur at specific points in time, called the optimal switching times. Thus we show that optimal controller signals can be parameterized with a small number of variables (switching times). (In Section 6.3, we work through an example of time-optimal control of a second-order, linear mechanical system.)

6.1.3 A NEURAL NETWORK CONTROL METHODOLOGY

In this chapter, we develop a methodology using feed-forward networks for open-loop control. In Section 6.4, we present a series of experiments to test this methodology. In the first experiment, we verify that a feed-forward network is capable of approximating the functional relationship, $t_s = f(x_0, x_f)$ (Section 6.4.2). In the second experiment, the networks are trained directly on the dynamical state trajectory (Section 6.4.2). Changes in switching times generated by the network can be evaluated by their effect on the state trajectory - without any prior knowledge of what the ideal switching times may be. Thus, under this control scheme, it is not necessary that we have a precise mathematical description of the dynamical system to be controlled (also referred to as the plant in control theory). However, back-propagation requires the calculation or estimation of the partial derivatives of the plant for the adjustment of the network weights. Since the chemotaxis algorithm adjusts weights randomly, these calculations are unnecessary. Finally, we report experiments on a preliminary attempt to apply this methodology to open-loop control of a two-link robotic arm (Section6.4.3).

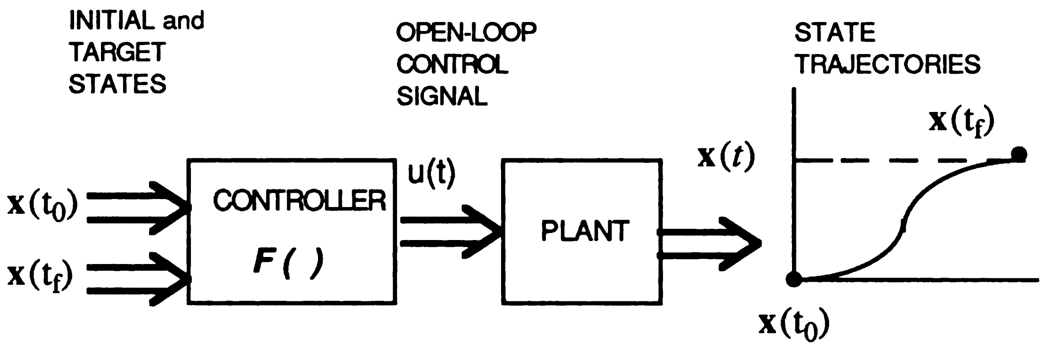


Figure 6.1: Schematic Representation of a Problem in Open-Loop Motor Control

The controller's function is to generate an appropriate control signal as a function of the initial and target states of the system to be controlled. We wish to employ a neural network to perform the function of the controller.

6.2 TIME-OPTIMAL CONTROL THEORY

6.2.1 PROBLEM STATEMENT

Consider an nth-order mechanical system with a bounded forcing function,

$$\dot{\mathbf{x}} = g(\mathbf{x},\mathbf{u})$$

$$|u(t)| \leq 1$$

where $\mathbf{x}(t)$ and $\mathbf{u}(t)$ are respectively the state and control variables. The task is to drive this system from its given initial state $\mathbf{x}(0) = \mathbf{x}_0$, to the final state $\mathbf{x}(t_f)$, in minimum time. This problem is routinely solved in control theory by maximizing a performance index

$$J = \int_0^{t_f} f_0\,(\mathbf{x},\mathbf{u},t)\,dt$$

where f_0 is a cost function. For time-optimal control, $f_0 = -1$ since maximizing the performance index gives the minimum time. Thus, the statement of the problem is:

$$\max_{\{|u_i| \leq 1\}} J = \int_0^{t_f} -1\,dt = -\,t_f$$

subject to the system constraints.

6.2.2 PONTRYAGIN'S MAXIMUM PRINCIPLE

Pontryagin's maximum principle (Pontryagin et al. 1962) gives a necessary condition that an optimal control must satisfy and has become the basic method for computing optimal controls (Intriligator 1971, Takashi et al. 1972). The maximum principle can be thought of as an extension of the method of Lagrange multipliers to dynamic optimization and control problems. The maximum principle involves the introduction of *n costate* (or adjoint) variables $y_i(t)$, $i=1,\dots,n$. A scalar function, called the Hamiltonian H, is then defined as the sum of the cost function plus the inner product of the vector of costate variables and the vector of functions defining the rate of change of the state variables:

$$H(\mathbf{x},\mathbf{u},\mathbf{y},t) = f_0(\mathbf{x},\mathbf{u},t) + \mathbf{y}\dot{\mathbf{x}} = f_0(\mathbf{x},\mathbf{u},t) + \mathbf{y}g(\mathbf{x},\mathbf{u})$$

Application of the maximum principle involves solving for the trajectories $\mathbf{u}(t)$, $\mathbf{y}(t)$, and $\mathbf{x}(t)$ satisfying

$$\max_{\{u \in \Omega\}} H(\mathbf{x},u,\mathbf{y},t) \quad \text{for all } t, \quad t_1 \leq t \leq t_f$$

where Ω is the class of permissible controls. The maximum principle states that the function $H(\mathbf{u})$ attains its maximum at $\mathbf{u} = \mathbf{u}^*$, where $\mathbf{u}^*$ represents the optimal control. Moreover, for the case of time-optimal control under this formulation, $H(\mathbf{u}^*) = 0$ (Leitmann 1981). The equations of motion for the costate and state variables respectively are:

$$\dot{\mathbf{y}} = -\frac{\partial H}{\partial \mathbf{x}} \quad , \quad \dot{\mathbf{x}} = -\frac{\partial H}{\partial \mathbf{y}}$$

These equations for the state and costate variables are posed as initial and final value problems, respectively.

6.2.3 BANG-BANG CONTROL

It is often possible to find important properties of the optimal control signal by inspection of the Hamiltonian. First, we note that if the Hamiltonian for a given problem can be decomposed into two functions, $H_1(\mathbf{u})$ which is a function of $\mathbf{u}$ and H_2 which is not a function of $\mathbf{u}$:

$$H(u) = H_1(u) + H_2 ,$$

we need only consider $H_1(\mathbf{u})$. Now the control variable **u** is usually restricted to some prescribed range ($\mathbf{u} \in \Omega$). If the optimal control signal takes on values for **u** inside this range, it is called an "interior maximum"; if the optimal control only takes on its limiting values of its allowed range, it is called a "boundary maximum". Through mathematical analysis of H_1, one can determine whether $\mathbf{u}^*(t)$ is an interior or boundary maximum over some or all of its trajectory. For example, in the case of a single control variable, if H_1 is proportional to u or

$$H_1(u) = g(t)\, u ,$$

u^* takes on its limiting values, u_{min} and u_{max} according to the following criterion:

$$\begin{aligned} u^* &= u_{max} \quad \text{if } g(t) > 0 , \\ u^* &= u_{min} \quad \text{if } g(t) < 0 . \end{aligned}$$

This mode of control is called *bang-bang control*, since the control variable takes on only its extremal values. The times at which u^* changes from one extreme to another are called the optimal *switching times*. The switching times correspond to the zero-crossings of g(t), which is commonly referred to as the *switching function*. In many cases, analysis of this function can be used to specify the form of the control signal.

In some cases, we can even determine the number of switching times in a bang-bang control signal. Control of a linear time-invariant (LTI) system is an interesting special case. If the optimal control of an LTI system can be shown to be bang-bang, under certain conditions, the number of switching times are specified by the following theorem:

Theorem. (Bellman, Glicksberg and Gross 1956) Consider the system:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$$

where **A** and **B** are constant matrices of dimension $n \times n$ and $n \times m$ respectively; the control constraint set is a parallelepiped in R^m; that is,

$$U = \left\{ u \in R^m \,\middle|\, u_k^{\min} \leq u_k \leq u_k^{\max},\ k = 1,2,\dots,m \right\}$$

and the cost function is $f_0(\mathbf{x},\mathrm{u},t) = 1$. If (*a*) the eigenvalues of the matrix **A** are real, and (*b*) the component $u_k(\cdot)$ of the control variable $u(\cdot) : [\,t_0,\ t_f\,] \in R^m$ is bang-bang, then there can be at most $n - 1$ switches in the value of $u_k(t)$.

A simple proof of this theorem can also be found in Leitmann (1981). Thus, if for a given system we can demonstrate that optimal control is bang-bang and specify the number of switching times, we have a complete parameterzation of the optimal control. In the following section we work through an example of time-optimal control of a second-order system. This system is then used for a test case of our neural network control methodology in Sections 6.4.1 and 6.4.2.

6.3 EXAMPLE: TIME-OPTIMAL CONTROL OF A LINEAR SECOND-ORDER SYSTEM

We consider a second-order linear time-invariant system with a bounded forcing function:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}u$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -k & -b \end{bmatrix};\ \mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix};\ \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$|u(t)| \leq 1$$

where $\mathbf{x}(t)$ and $u(t)$ are respectively the state and control variables. The task is to drive this system from its given initial state $\mathbf{x}(0) = \mathbf{x}_0$, to the final state $\mathbf{x}(t_f) = [x_f, 0\,]^T$, in minimum time.

As before, the performance index for time-optimal control is $f_0 = -1$. Thus, the statement of the problem is:

$$\max_{\{|u| \leq 1\}} J = \int_0^{t_f} -1 \, dt = -t_f$$

subject to the system constraints.

The Hamiltonian for this example is

$$\begin{aligned} H &= f_0 + \mathbf{y}\dot{\mathbf{x}} \\ &= -1 + \mathbf{y}(\mathbf{A}\mathbf{x} + \mathbf{b}u) \\ &= -1 + y_1 x_2 - k y_2 x_1 - b x_2 y_2 + u y_2 \end{aligned}$$

Since the control variable u occurs only in the final term, to maximize H we need to choose u(t) such that the term uy_2 is maximized. Therefore, u(t) should be at its highest possible positive value (+1) when y_2 is positive and its maximal negative value (-1) when y_2 is negative, or

$$u^*(t) = \text{sgn}\{y_2(t)\}.$$

Thus, control of this system is ***bang-bang,*** and the function $y_2(t)$ is the corresponding switching function because it changes sign when the optimal control switches from one extremal value to the other.

6.3.1 CALCULATION OF THE SWITCHING FUNCTION

In general, there is no closed form solution for the switching function, $y_2(t)$. It is, however, relatively easy to find the solution through numerical methods (at least for $n = 2$). An insight into the nature of the difficulty can be obtained by considering the equations of motion for the co-state vector in the specific case of a second-order system:

$$\dot{y}_1 = -\frac{\partial H}{\partial x_1} = ky_2$$
$$\dot{y}_2 = -\frac{\partial H}{\partial x_2} = -y_1 + by_2$$

The solution of this system is:

$$y_1(t) = C_1 e^{-r_1 t} + C_2 e^{-r_2 t}$$

$$y_2(t) = \frac{-r_1}{k} C_1 e^{-r_1 t} - \frac{r_2}{k} C_2 e^{-r_2 t}$$

where

$$r_{1,2} = \frac{-b \pm \sqrt{b^2 - 4k}}{2}$$

However, we cannot determine the arbitrary constants since only the final values are specified for the costate variables. Nevertheless, there is some hope. We can get one constraint on these constants from the Hamiltonian. At $t = t_0$ or t_f, the Hamiltonian is both maximized and equal to zero (Leitmann 1981). Further, at time $t = 0$ we have the boundary conditions $x_1(0) = 0$ and $x_2(0) = 0$. Also, since the plant must initially be accelerated, $u^*(0^+) = +1$. Therefore solving for the Hamiltonian at $t=0^+$ yields one constraint on the two variables.

$$\max H\big|_{t=0} = -1 - \frac{r_1}{k} C_1 - \frac{r_2}{k} C_2 = 0$$

or

$$C_1 = -\frac{1}{r_1}(r_2C_2 + k)$$

Let $t = t_s$ be the time at which the switching function $y_2(t)$ changes sign (or $y_2(t) = 0$). Setting y_2 equal to zero at the switching instant we obtain an expression for C_2:

$$y_2(t_s) = \frac{1}{k}(r_2C_2 + k)e^{-r_1t_s} - \frac{r_2}{k}C_2e^{-r_2t_s} = 0$$

or

$$C_2 = \frac{ke^{-r_1t_s}}{r_2\left[e^{-r_2t_s} - e^{-r_1t_s}\right]}$$

So, if t_s is known, we would be able to fix the other constant. The boundary conditions at t_f yield no additional constraints on C_1 or C_2 since they introduce the additional variable t_f.

Figure 6.2a shows y_2 and u^* for $t_s = 1$. If r_1 and r_2 are real, the switching function $y_2(t)$ could change sign only once over the interval $0 \le t \le t_f$; therefore, there is only one switching time. (This is simply a special case of the theorem in the previous section.)

It is useful to reinforce that application of the the maximum principle gives two important characteristics of the optimum solution: (a) time-optimal control is bang-bang, and (b) For r_1 and r_2 real, there is at most one switching time t_s in the optimal control trajectory $u^*(t)$ between its two extremal values. We use this knowledge, in the next section, to solve the dynamical equations of the system.

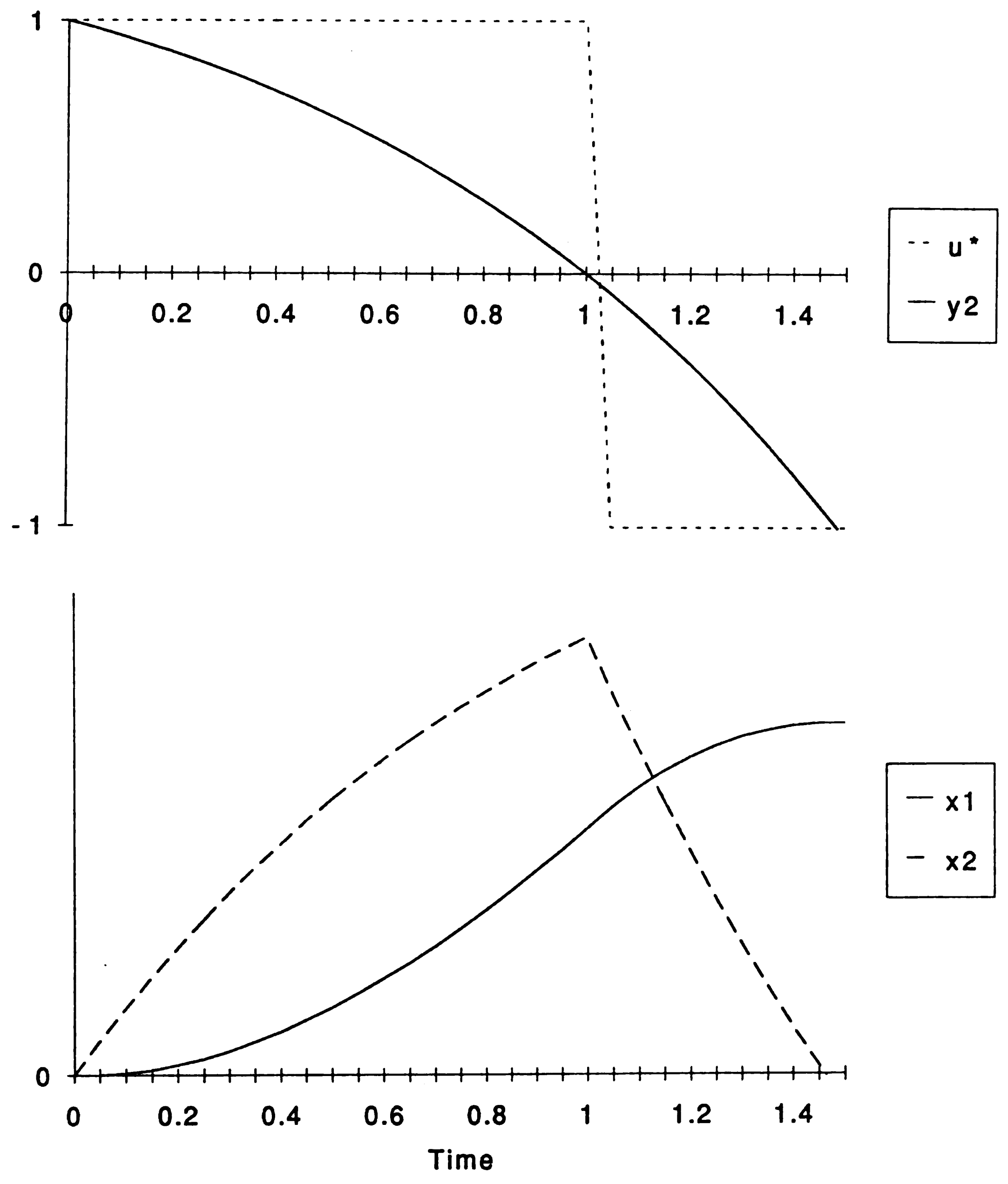


Figure 6.2 a & b: Time Optimal Control of a Second-Order, Linear System:
Figure 6.2 a shows the trajectory of the costate variable y_2 (also the switching function) and the optimal, 'bang-bang', control strategy. Note that the control variable changes sign at the same time ($t_S = 1$) as y_2. Figure6.2 b shows the state trajectories x_1 (position), and x_2 (velocity). The vertical axes in both figures are dimensionless units.

6.3.2 OPTIMAL CONTROL WITH A SINGLE SWITCHING TIME

We know that the optimal control trajectory is:

$$u^*(t) = \begin{cases} +1 & 0 \le t \le t_s^* \\ -1 & t_s^* \le t \le t_f \end{cases}$$

where t_s^* is the optimal switching time. Now the equations of motion can be solved. The homogeneous solution of the dynamical system is

$$x_1(t) = C_1 e^{-r_1 t} + C_2 e^{-r_2 t}$$

where

$$r_{1,2} = \frac{+b \pm \sqrt{b^2 - 4k}}{2}$$

For $0 \le t \le t_s^*$, the particular solution $x_1(t)$ is determined from the initial conditions $\mathbf{x}(0) = \mathbf{0}$. Combined with the forcing function $u^*(t) = 1$, we get

$$x_1(t) = \frac{1}{k} + \frac{1}{k(r_1 - r_2)}\left[-r_2 e^{-r_1 t} + r_1 e^{-r_2 t}\right]$$

For $t_s^* \le t \le t_f$, the particular solution $x_1^{\sim}(t)$ is determined from the final conditions, namely $x^{\sim}_1(t_f) = x_f$, $x^{\sim}_2(t_f) = 0$:

$$\tilde{x}_1(t) = -\frac{1}{k} + \frac{(x_f + \frac{1}{k})}{(r_1 - r_2)}\left[-r_2 e^{-r_1(t - t_f)} + r_1 e^{-r_2(t - t_f)}\right]$$

We can now match the two solutions at the switching instant $t = t_s^*$.

$$x_1(t_s^*) = \tilde{x}_1(t_s^*)$$
$$x_2(t_s^*) = \tilde{x}_2(t_s^*)$$

These two equations define one more set of boundary conditions. Imposing these conditions leads us to the pair of transcendental equations:

$$2 + \frac{r_2 e^{-r_1 \dot{t}_s}}{(r_1 - r_2)}\left[1 + (kx_f + 1)e^{r_1 t}\right] = \frac{r_1 e^{-r_2 \dot{t}_s}}{(r_1 - r_2)}\left[1 + (kx_f + 1)e^{r_2 t}\right]$$

$$e^{-r_1 \dot{t}_s}\left[1 + (kx_f + 1)e^{r_1 t}\right] = e^{-r_2 \dot{t}_s}\left[1 + (kx_f + 1)e^{r_2 t}\right]$$

These equations have no closed form analytical solution (like linear or quadratic equations). Their roots can be found numerically by iterative methods, such as Newton's method. For a single variable (single switching time), Newton's method is unproblematic. For larger dimensions ($n > 2$), Newton's method requires repeated inversion of matricies (Strang 1980). Complications arise when during the numerical iteration, singular or ill-conditioned matricies occur.

Further complications arise when the system dynamics are nonlinear. Iterative numerical methods often become unstable and fail to converge on these problems since they are often based on local error estimates (such as gradients) which do not predict global solutions. In this case, we must resort to global optimization algorithms which cannot be guaranteed to converge (Cutler 1988). Many methods have been used to optimize switching times in bang-bang control of nonlinear systems. Lehman has applied the Bremermann optimizer (Bremermann 1970) successfully for time-optimal control of nonlinear extraocular muscle-plant models (Lehman and Stark 1979;83). Masri, Bekey and Stafford (1980) applied a stochastic optimization procedure to optimize switching times for a five-degrees-of-freedom system.

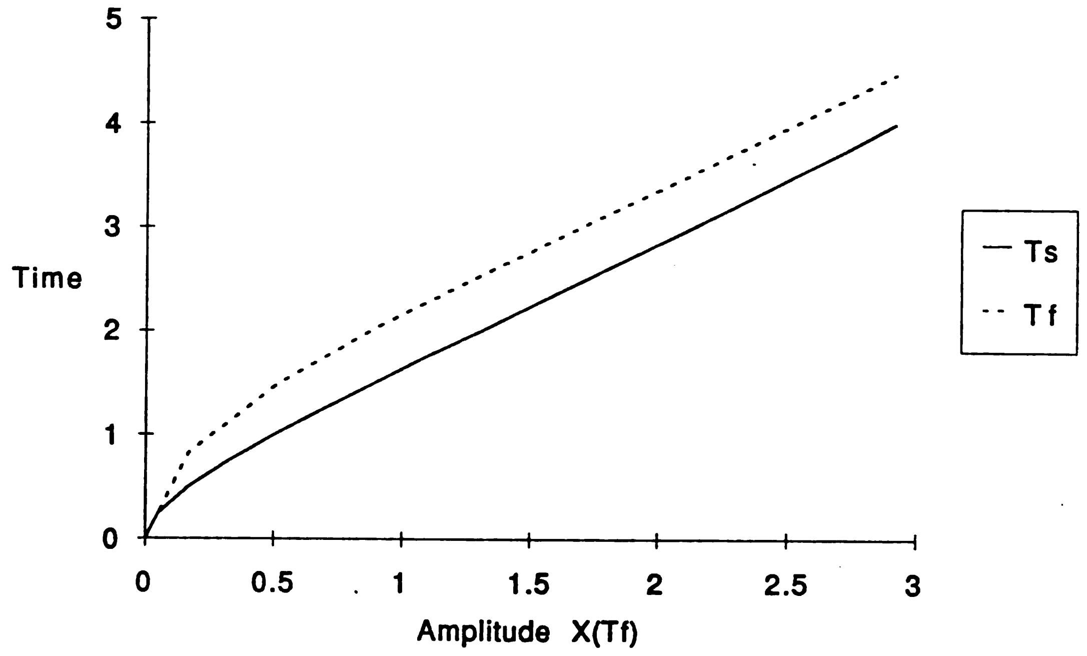


Figure 6.3: Optimal switching and stopping times.
Time-optimal control of an overdamped system at various amplitudes. Notice that the ratio ($R = t_s/t_f$) approaches 1 for larger amplitude movements. [System parameters: m=1, b=1, k=0.1].

6.3.3 SWITCHING TIMES AS A FUNCTION OF SYSTEM PARAMETERS

Now let us consider the problem of controlling of a pure inertial object (i.e., $b = 0$, $k = 0$ in the dynamical equations). The time-optimal control strategy would be to accelerate the mass for half of the trajectory and decelerate for the second half. Thus, $t_s/t_f = 1/2$. The amplitude of the movement would vary with t^2. At the other extreme, a pure viscoelastic system with no inertia ($m = 0$) would not require any braking at all. The optimal strategy in this case would be simply to drive the system with maximal input until it arrives at the final position. Thus, we have arrived at upper and lower bounds for the ratio $R = t_s/t_f$ ($0.5 \leq R \leq 1.0$). Figure 6.3 shows how the values of t_s and t_f change with amplitude for an overdamped system (model parameters: $m = 1$, $k = 0.1$, $b = 1$). For small amplitude movements, system inertia dominates and R approaches 1/2. For larger amplitudes, viscoelastic forces dominate. For critically damped systems or very large amplitude movements, there may be no braking pulse in the optimal control signal at all ($t_s \rightarrow \infty$ or $t_s \rightarrow t_f$).

6.4 NEURAL NETWORKS FOR OPEN-LOOP CONTROL

6.4.1 TRAINING A NEURAL NETWORK TO GENERATE SWITCHING TIMES

In this section, we describe the training of neural networks to approximate a class of functions of the form:

$$t_s^* = f\,\{x(t_0), x(t_f)\}$$

for the second-order system described in the previous section.

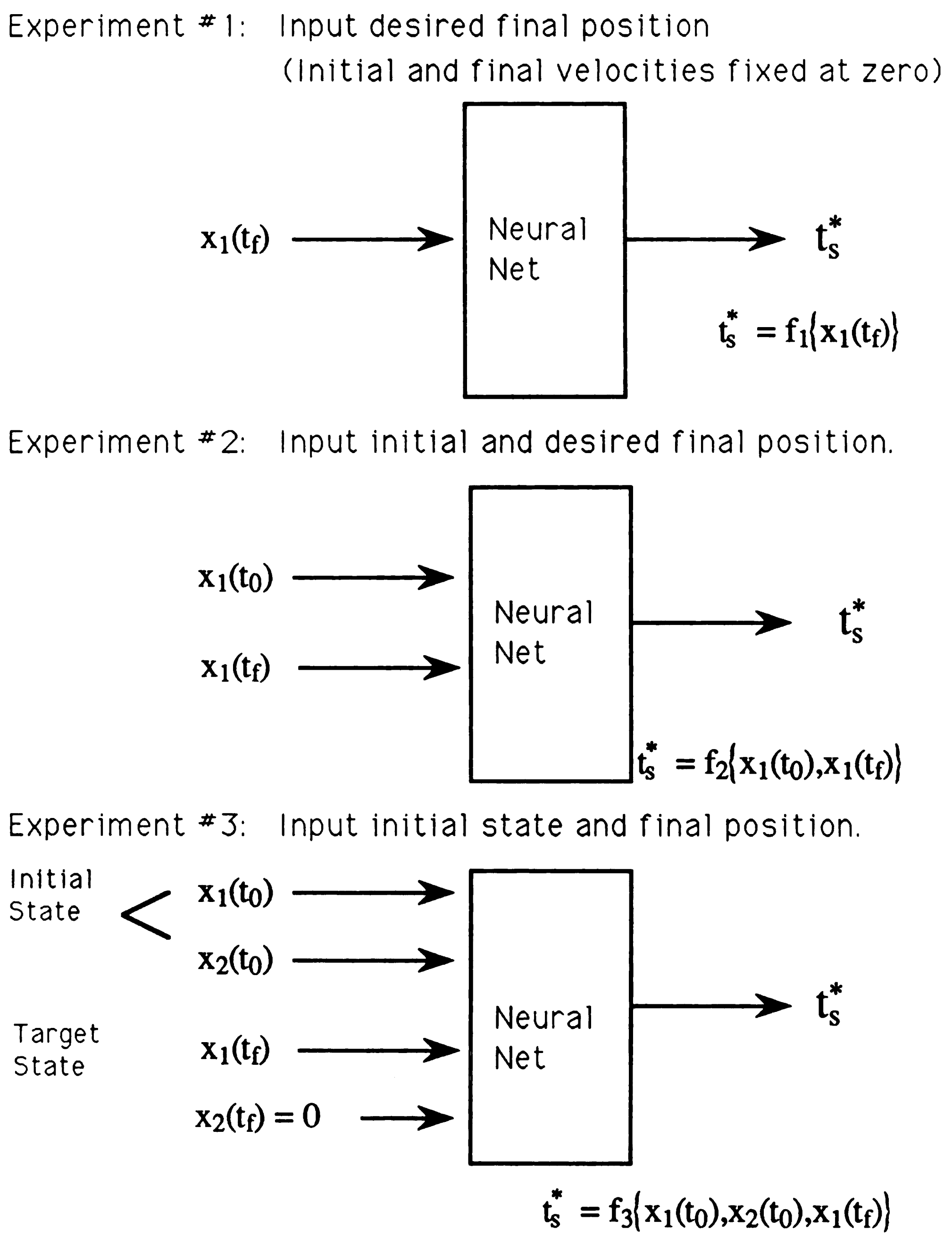

Experiment #1: Input desired final position
(Initial and final velocities fixed at zero)
$x_1(t_f)$
Neural Net
t_s^*
$t_s^* = f_1\{x_1(t_f)\}$
Experiment #2: Input initial and desired final position.
$x_1(t_0)$
$x_1(t_f)$
Neural Net
t_s^*
$t_s^* = f_2\{x_1(t_0), x_1(t_f)\}$
Experiment #3: Input initial state and final position.
Initial State
$x_1(t_0)$
$x_2(t_0)$
Target State
$x_1(t_f)$
$x_2(t_f) = 0$
Neural Net
t_s^*
$t_s^* = f_3\{x_1(t_0), x_2(t_0), x_1(t_f)\}$

Figure 6.4: Schematic Representation of Three experiments

Network inputs consisted of the target, final state and the initial system state. The network output is the corresponding optimal switching time. Experiments 1-3 are increasing generalizations of the problem.

Figure 6.3 illustrates the less general relationship:

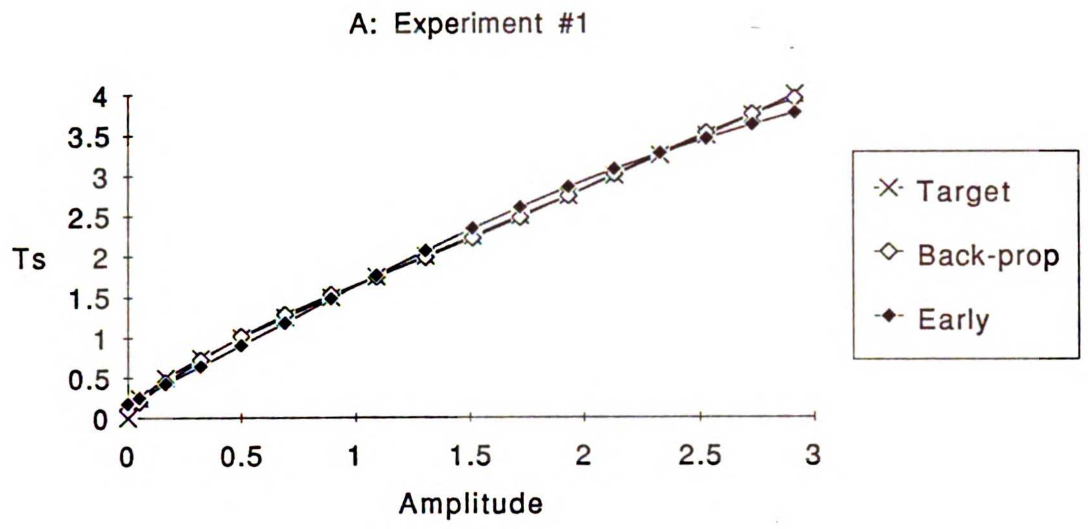
$$t_s^* = f_1\{x_1(t_f) \mid x_1(t_0) = x_2(t_0) = x_2(t_f) = 0\}$$

We can see that, in this case, this transcendental relationship can be represented as a continuous function. According to the theoretical results of Cybenko (see Chapter 3), neural networks can approximate any continuous function. However, these theoretical results do not predict how many units or how much training would be required. This section describes a series of three experiments to demonstrate the effectiveness of the back-propagation and chemotaxis training algorithms in finding *practical* neural network solutions of this task.

The networks are trained to generate the time-optimal switching time, given the initial and target states of the system as inputs. The three experiments are illustrated schematically in Figure 6.4. In the first experiment, the functional relationship is identical to the example problem in the previous section (illustrated in Figure 6.3). The system is initially at rest at the origin and is to be driven to a target state, that is $\{x = (0,0) \rightarrow (x_f,0)\}$. In the second experiment, initial and final positions are given as inputs, that is $\{(x_0,0) \rightarrow (x_f,0)\}$. Finally, in the third experiment, the initial state (position and velocity) and the target position are inputs - making the switching time a function in three dimensions, that is $\{\mathbf{x}_0 = (x_0,v_0) \rightarrow (x_f,0)\}$. All three of these tasks represent real-world tasks in control theory (drive a system to a new position and bring it to rest in minimum time). A fourth experiment, where the target state has nonzero velocity is also possible, but it was not attempted.

6.4.1.1 Experiment #1

The training set consisted of 17 sampled input/output pairs spanning 'target' amplitudes from 0 to 3 (see Figure 6.5a). To create the training set, the optimal switching times were found numerically.



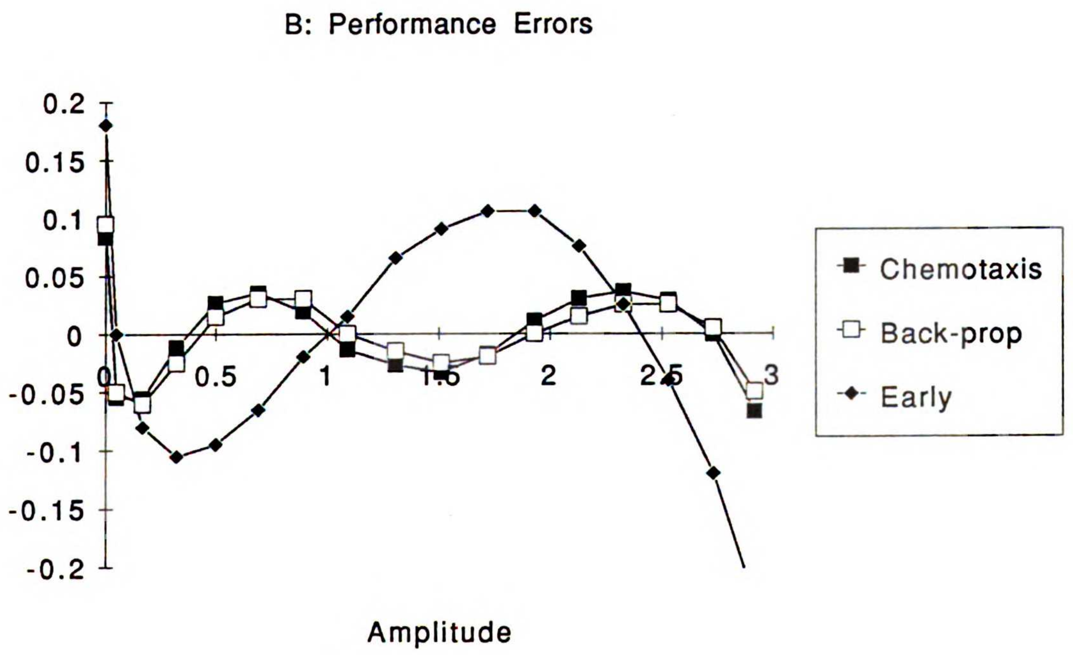


Figure 6.5a&b: Performance of a Trained Network in Experiment #1. Figure 6.5a shows the performance of a network trained by back-propagation as compared to optimal switching times. The training set samples the dynamic range of the system. The input/output pairs used in the training set are the shown with markers. Figure 6.5b shows the error magnitudes as a function of movement amplitude (x_f) for both training algorithms as well as the sub-optimal performance of a network early in training.

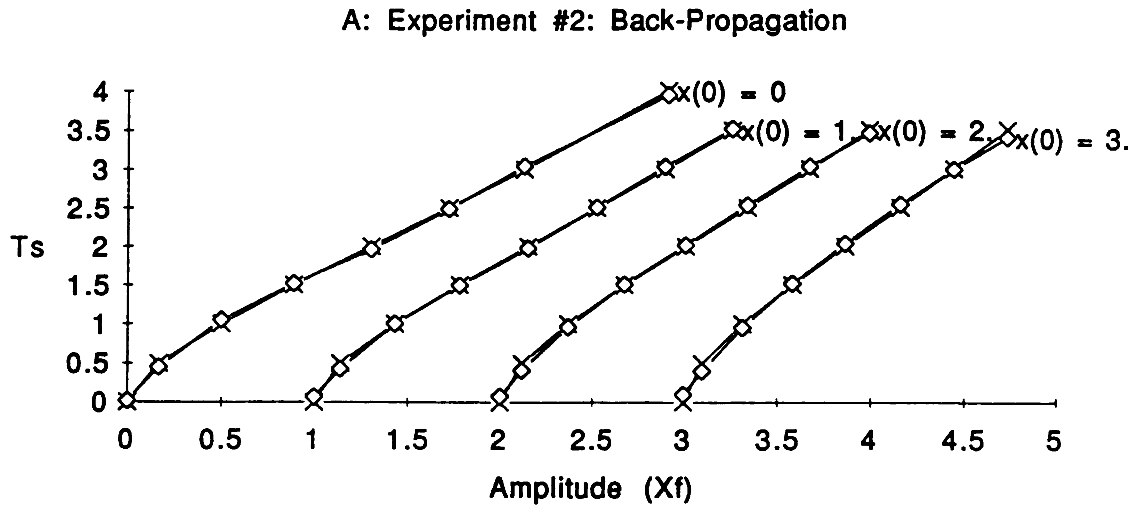
The network consisted of a single input node (i.e. the target state), 15 hidden units and a single output unit (1-15-1).

Networks trained with the back-propagation algorithm were trained until performance improvement stalled (i.e., no improvement for 100 cycles). The back-propagation runs averaged about 4000 cycles. Networks trained with the chemotaxis algorithm were trained for a fixed 6000 cycles, although performance continued to show slight improvement with continued training beyond that point. As was pointed out in Chapter 2, the chemotaxis algorithm requires *half* as much computation per training cycle as back-propagation. Performance of the two trained networks is shown in Figure 6.5 a&b. The performance of the two networks was comparable.

Inspection of training progress sheds some light on how neural networks learn to approximate functions. Figure 6.5 shows the performance of a network midway through training (the trace labelled "Early"). Improvement in performance of the network at that stage in training was slow suggesting the presence of a possible "saddle point" in the "energy" landscape. The performance of this sub-optimal network represents a kind of 'least-squares' fit of a single sigmoid function to the data. Later in training, we can see that higher order corrections are generated by the network, indicating that the network solution now represents a functional approximation by superposition of *several* sigmoid functions.

6.4.1.2 Experiment #2

Now, the network is given two inputs - the target position (as in Experiment #1) as well as the initial position. the training set consisted of 32 input/output patterns with x_0 ranging from 0 to 3 and x_f ranging from 0 to 5. The initial velocity was set to zero. The network architecture was 2-15-1. Final network performance of networks trained with back-propagation and chemotaxis is given in Table 6.1 and graphically in Figure 6.6a-b.



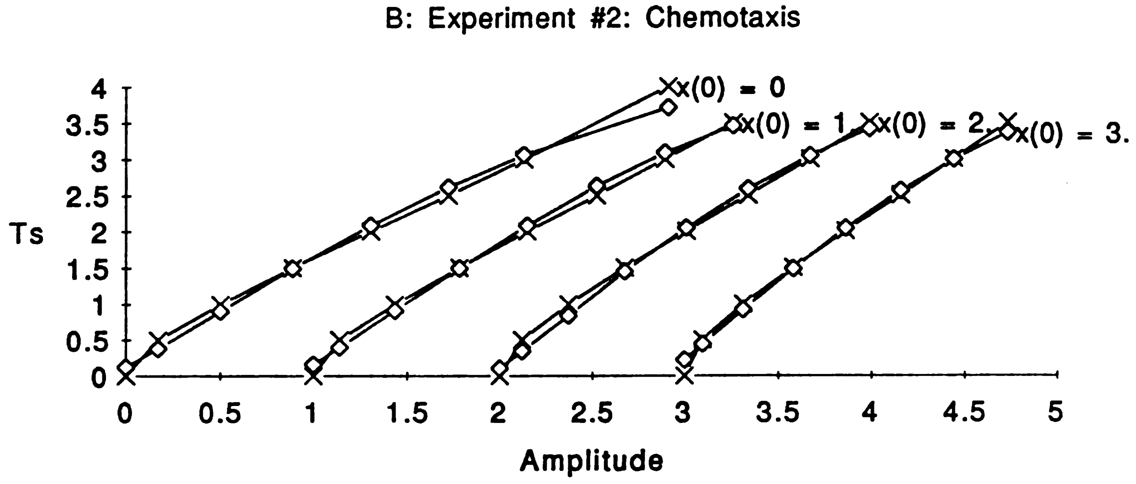


Figure 6.6 a&b: Network Performance for Experiment #2.
Training set for this more general case consisted of 32 patterns spanning the dynamic range.

6.4.1.3 Experiment #3

Here the entire initial state (position and velocity) as well as the target position were used as inputs. Initial conditions were chosen over a specified range ($x_2(t_0)$:[0,1],$x_1(t_0)$:[0,3]). The training set consisted of 112 sample input/output patterns. The network architecture was 3-20-1. Figure 6.7 shows performance of the trained network on sample patterns from the entire training set (A full representation would require a plot of a function of three variables).

6.4.1.4 Conclusions

We have shown that it is not only theoretically possible for a neural network to estimate the functional relationships described in Section 6.3, but it is *practical* to train a relatively small network to do so with **either** the back-propagation **or** chemotaxis training algorithms. Although the performance of networks trained with the chemotaxis algorithm was generally a little inferior to those trained with back-propagation, the chemotaxis algorithm offers some important computational advantages. The most important feature (for our purposes) is that it does not require gradient calculations.

In the next section, rather than training networks to estimate *known* switching times, we train a network to generate switching times as intermediate variables which, in turn, are used to control an unknown dynamical system. In order to calculate weight changes using the back-propagation algorithm, we would need to estimate the quantity $\partial E/\partial \tau^*$ (where E is the state trajectory error), which could become very expensive, computationally.

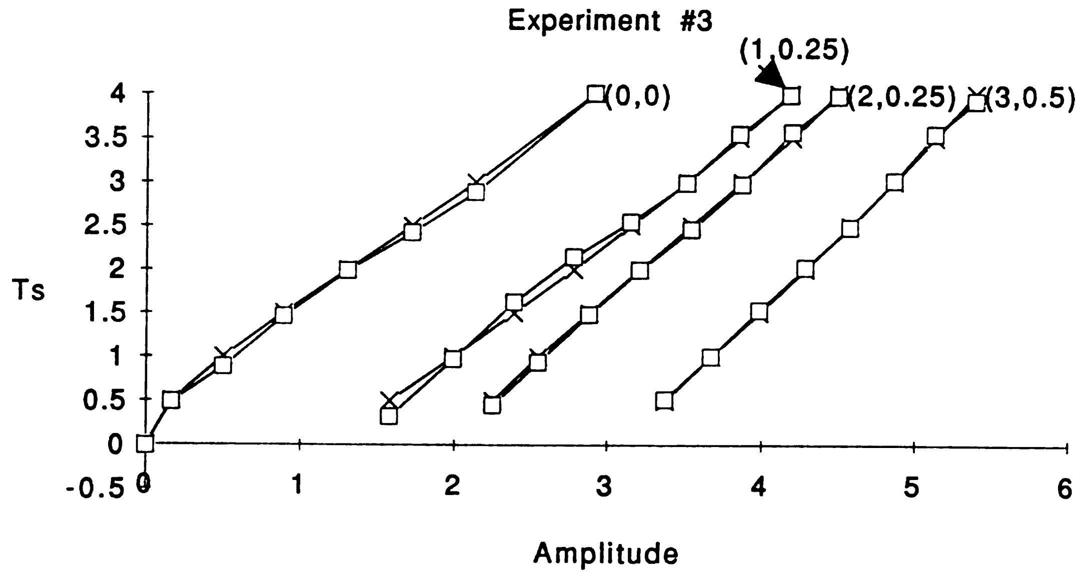


Figure 6.7: Network Performance for Experiment #3.
Only a subset of the 112 patterns in the training set are shown. A Complete representation would require a four-dimensional plot.

Table 6.1: Summary of Results for Section 6.4.1*

Expt. #	Network Inputs	Training Set Size	Average Error/ Pattern	
			Chemotaxis	Back-Propagation
1	Xo	17	0.00148	0.00135
2	Xo, Xf	32	0.0121	0.00245
3	Xo, Vo, Xf	112	0.0154	0.00742

*Networks were trained to generate the optimal switching times. Back-propagation algorithm was implemented on SunNet software. Algorithm parameters: learning rate 0.05, momentum term 0.95 to 0.99.

6.4.2 TRAINING NEURAL NETWORKS TO CONTROL A SYSTEM WITH UNKNOWN SWITCHING TIMES

In real-world motor control problems, the system (plant) dynamics are unknown. Controlling a "black box" is not, however, a hopeless task. One can evaluate the effectiveness of various control signals by observing the dynamical trajectories at the output.

In the following experiments, we drive an unknown plant with a neural net and use the resultant state *trajectory* as a measure of network performance. The methodology of these experiments is illustrated in Figure 6.8. As in Section 6.4.1, a neural network takes as inputs the initial and target system states and generates a corresponding switching time. A signal generator (SG) translates the switching time into the corresponding control signal u(t) which is used to drive the plant directly. The output of this system is simply the observed state trajectory. Notice that no knowledge of the plant or its optimal switching times is necessary for this control scheme. For this preliminary study, we used the same second-order plant described in Section 6.3 allowing for validation of the method.

To achieve this objective, we are presented with two technical problems:

1.) Given the state trajectory, how should we measure the system performance?

2.) After performance is measured, how should the network weights be modified?

We want to drive the plant to its target state while minimizing transit time. We know the optimal controller signal for this system has only one switching time. This suggests a simple method for evaluating system performance. Referring to Figure 6.9, assume the heavy line represents the optimal trajectory. Non-optimal switching times will draw the actual trajectory off this course. If the switching time generated by the network is less than the optimal, the plant will

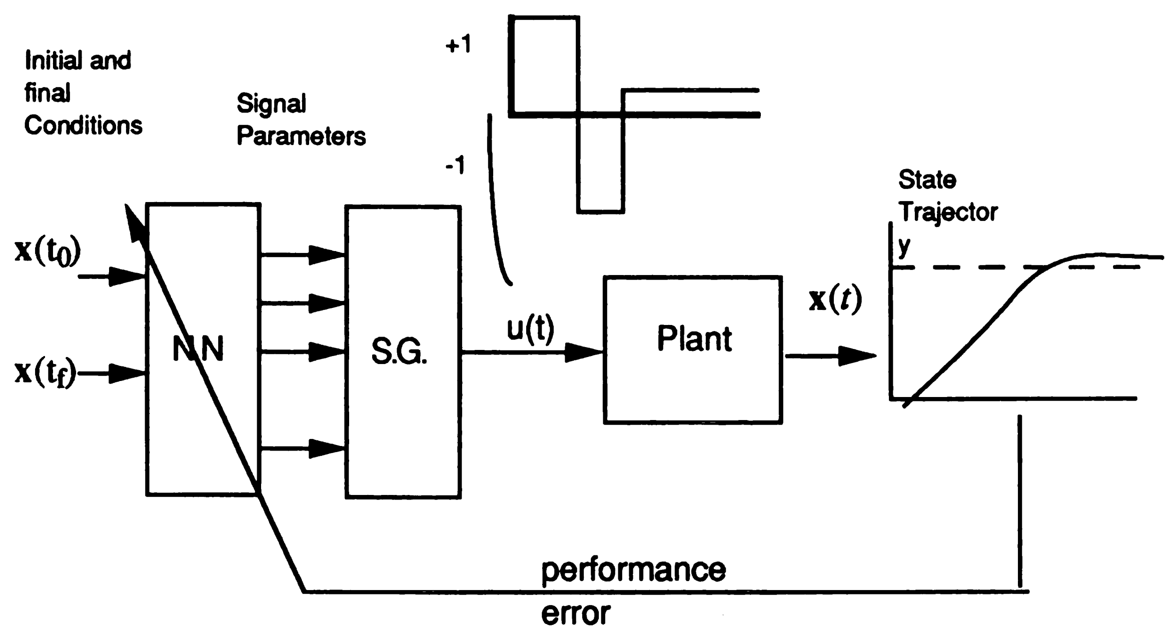
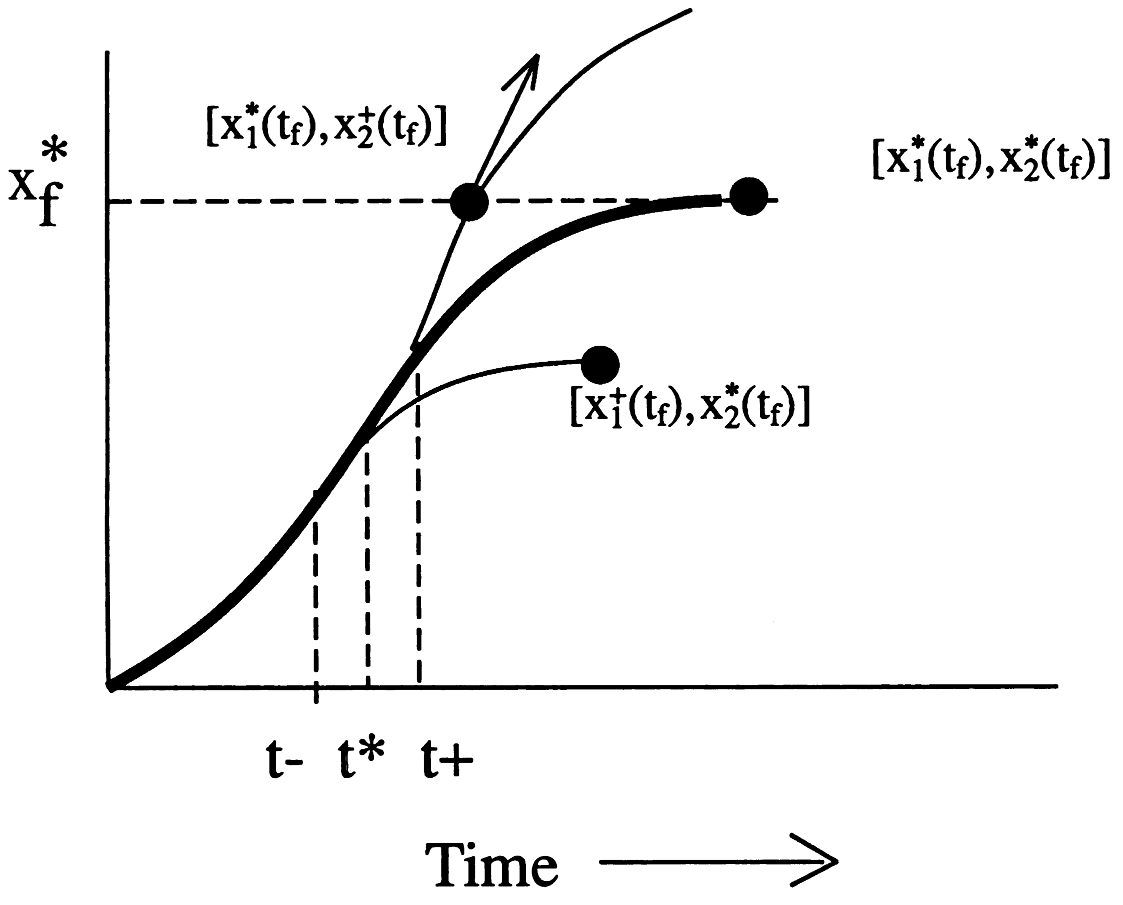


Figure 6.8: Experiment Design Used in Sections 6.4.2-6.4.3
The neural network (NN) receives initial and target state as inputs and generates switching times (signal parameters) as outputs. A signal generator (S.G.) generates the corresponding temporal control signal [u(t)] used to drive the plant. The state trajectory [x(t)] is then used to evaluate network performance (Figure 9) and modify the network weights.



x_f^*
$[x_1^*(t_f), x_2^+(t_f)]$
$[x_1^*(t_f), x_2^*(t_f)]$
$[x_1^+(t_f), x_2^*(t_f)]$
t- t* t+
Time

$$\text{for } t^+,\ e_{x_2} = (x_2^+ - x_2^*)$$

$$\text{for } t^-,\ e_{x_1} = (x_1^- - x_1^*)$$

Figure 6.9: Evaluation of System Performance

The dark trace represents the time-optimal trajectory to [X1(tf),X2(tf)]. If the network-generated switching time is short of optimal, the trajectory will come up short of optimal (X1-). If the switching time is larger than ts, the trajectory will have excess velocity when it reaches the target position (X2+). This suggests a simple method of measuring system performance.

come to rest short of the desired target position ($x_1^- < x_1^*$). On the other hand, if t_s is too large, when the system reaches the target position, it will have excess velocity ($x_2^+ > x_2^*$). Thus, when either the target position or velocity are attained, the error in the other state at that time can be measured. The performance error is defined as:

$$\text{Error} = \sum_{n=1}^{P} \left\{ (x_{2f}^+ - x_{2f}^*)^2 + (x_{1f}^- - x_{1f}^*)^2 \right\}$$

where P is the number of patterns in the training set.

The second problem arises when we realize that if we are working with an unknown plant, we cannot analytically calculate the state derivatives needed to apply the back-propagation algorithm. Psaltis has used numerical differentiation to estimate the local partial derivatives using small variations about the operating point for neural network control (Psaltis 1990). This method can become computationally intensive, especially with higher dimensional systems corrupted by noise. We propose a simpler method which does not require partial derivatives at all. That is, we apply the chemotaxis algorithm to the entire system.

In the previous section, we had demonstrated that the chemotaxis algorithm is capable of generating the switching times as a function of initial and final system states. The accuracy of networks trained with the chemotaxis algorithm was nearly as good as those trained back-propagation.

6.4.2.1 Simulation Results with a Second-Order Plant

The three experiments described in Section 6.4.1 were repeated using this new scheme. Initial experiments demonstrated that a few minor modifications in network architecture significantly improved the system performance. First, the chemotaxis algorithm seems to perform better when two layers of hidden units are used.

We think this is because it increases the dimensionality of the search space allowing several routes to a 'good' solution. This empirical result is supported by the theoretical analysis of Chester (1990a), which was reviewed in Chapter 3 (Section 3.2). Secondly, at the output layer, we used a simple piecewise linear unit rather than a sigmoid to avoid saturation effects.

Figure 6.10 shows the switching times (now intermediate variables in the system model) generated by the network as a function of movement amplitude for Experiment #1. Figure 6.11 and Figure 6.12 show performance of networks on Experiments #2 and 3, respectively. The results are summarized in Table 6.2.

6.4.2.2 Discussion

We note that since this approach measures state trajectory error instead of error in the switching time, we have implicitly included system *sensitivities* to variation in switching time. Modest variations in the switching time do not directly translate into significant errors in the final state. Therefore, in general, we should not expect the networks trained by this scheme to generate precisely the same switching times as those in Section 6.4.1. The system seems to be most sensitive to switching times in intermediate amplitude movements.

Table 6.2: Summary of Results for Section 6.4.2*

Expt. #	Network Inputs	Training Set Size	Network Architecture	Average Error/ Pattern
1	Xo	17	1-12-12-1	0.0084
2	Xo, Xf	32	2-10-10-1	0.033
3	Xo, Vo, Xf	112	3-15-8-1	0.035

*The network was trained on the state trajectory error. The error in this table is measured by the squared error in the optimal switching times, which is now an intermediate variable.

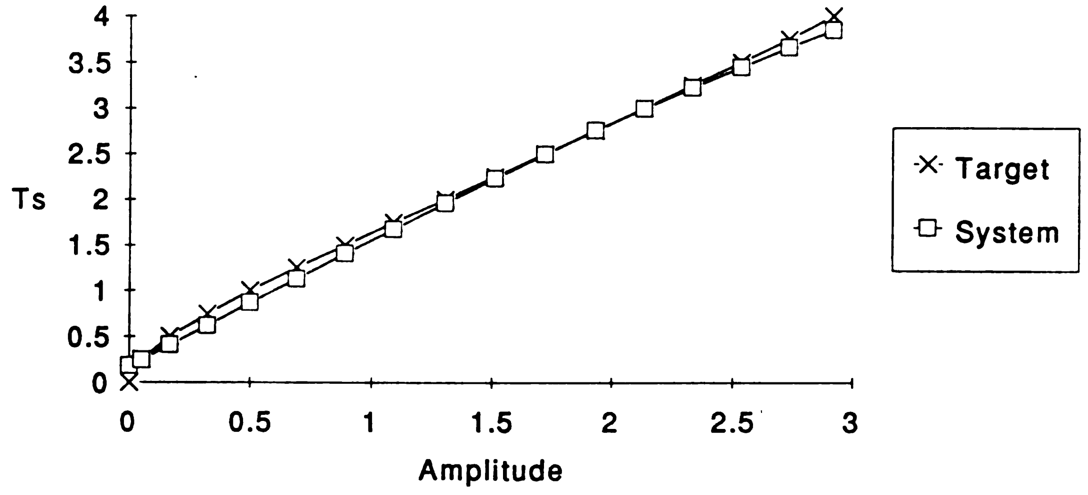


Figure 6.10: System Performance for Experiment #1.

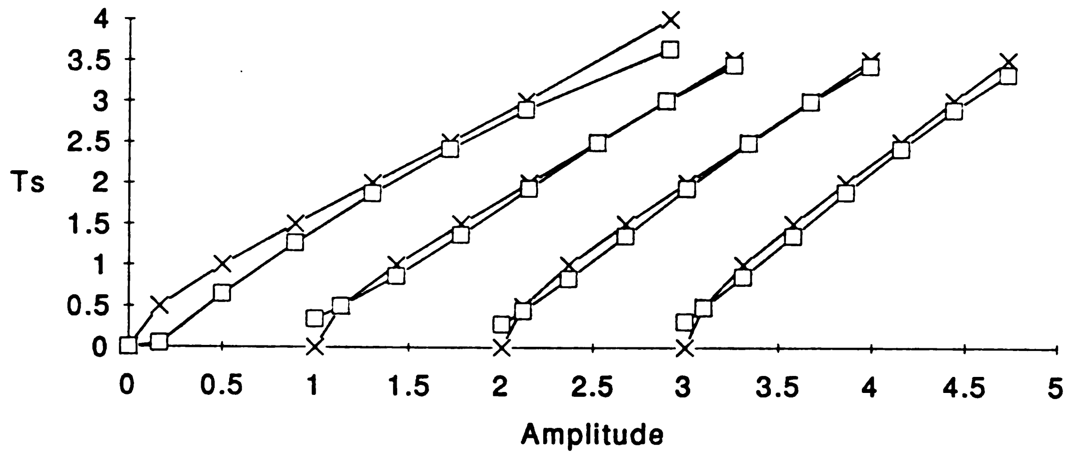


Figure 6.11: System Performance for Experiment #2.

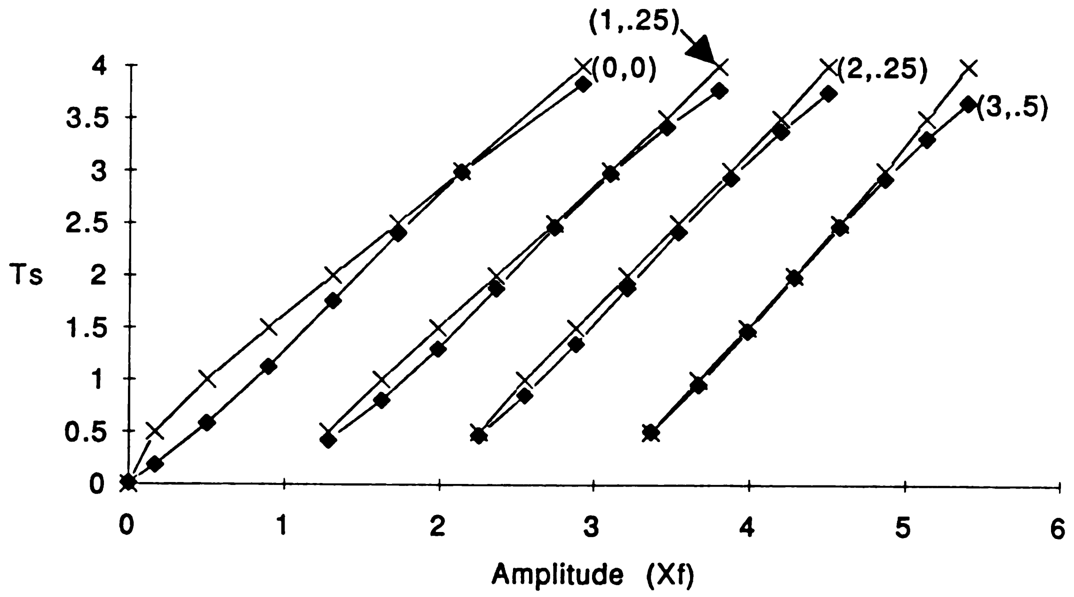


Figure 6.12: System Performance for Experiment #3.

6.4.3 PRELIMINARY RESULTS WITH A MULTILINK ARM

Next we wish to expand this methodology to the control of more complex plants. Time optimal control of higher dimensional systems (sometimes called high-*order* systems) involves multiple switching times. In most problems of control, exact system models are not available, and even when they are, calculation of the optimal controls can become a computationally intensive task. For example, to estimate the optimal switching times for a sixth-order nonlinear plant, Lehman and Stark (1979) used a stochastic optimization algorithm (Bremermann 1970) which required the use of a parallel computer. In the following we report the results of an attempt to apply our methodology to a nonlinear system with two control inputs.

6.4.3.1 Dynamics of a Planar Manipulator

We consider the control of a planar robotic arm with two links of equal length (see Figure 6.13). There have been several studies training neural networks to solve the inverse *kinematics* of robotic arms (e.g. Lee et al. 1990, Jack et al. 1990, Jorgensen 1990a,b). Kinematics deals with the basic geometry of the linkages. The inverse kinematics problem for a two-link arm is simply to derive the joint angles from the Euclidian (x,y) coordinates of the manipulator's end-point position. While the inverse kinematics of this system are analytically relatively simple, the inverse dynamics are not.

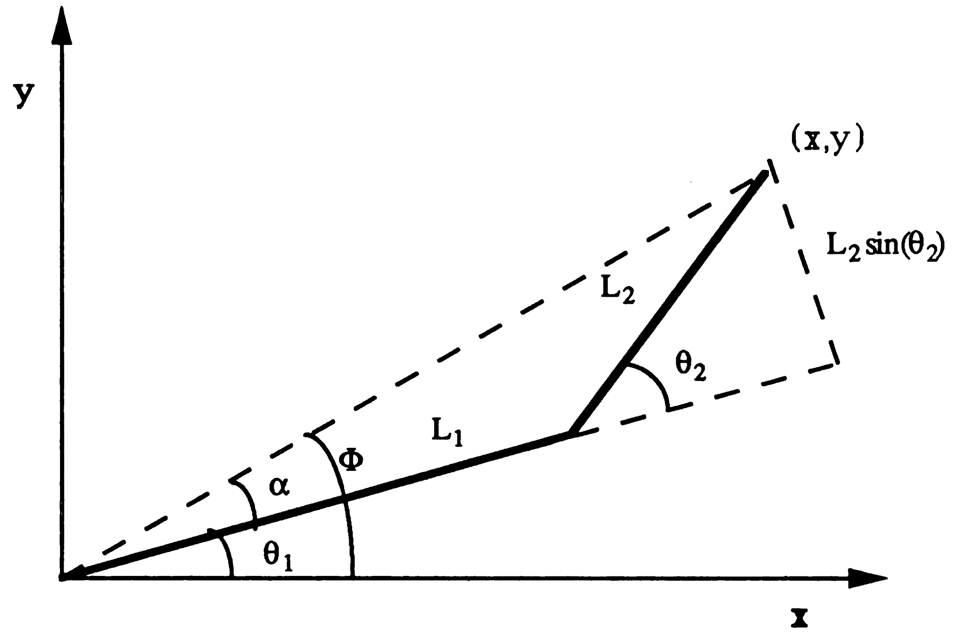


Figure 6.13: Geometry of a two-link, planar manipulator

The system considered in this study was purely inertial with no gravity or dynamical load. We assume two torque generators located at the "shoulder" (τ_1) and "elbow" (τ_2). The joint angles are denoted by θ_1 and θ_2 (Figure 6.13). The corresponding dynamical equations are given below (Horn 1974, Uicker 1967):

$$\ddot{\theta}_1 = \left[2/3\, T_1' - (2/3 + \cos(\theta_2))\, T_2'\right] / \left(16/9 - \cos^2(\theta_2)\right)$$

$$\ddot{\theta}_2 = \frac{\left[-(2/3 + \cos(\theta_2))\, T_1' + 2\left(5/3 + \cos(\theta_2)\right) T_2'\right]}{\left(16/9 - \cos^2(\theta_2)\right)}$$

where

$$T_1' = \tau_1 / (1/2\, ml^2) + \sin(\theta_2)\, \dot{\theta}_2 (2\dot{\theta}_1 + \dot{\theta}_2)$$

$$T_2' = \tau_2 / (1/2\, ml^2) - \sin(\theta_2)\, \dot{\theta}_1^2$$

In this example, the link mass (m) and length (l) were set to unity.

For this plant, the time-optimal switching times are unknown (Kahn 1969). Furthermore, since the system is purely inertial, we can expect that the system eigenvalues about *any* operating point are not all real valued. Consequently, the theorem in Section 6.2.2 does not strictly apply. Nevertheless, to a first approximation, we chose to try control with the minimum number of switches prescribed by the theorem. The dimension of the plant is 2, and there are two control variables; Thus, we allowed only one switching time per control input. In future work, we would like to increase the number of switching times to try to increase performance.

We have several choices available for the representation of network inputs and the measurement of performance. For example, since the length of each link is fixed, we could use the joint angles (θ_1, θ_2) or the Euclidian coordinates of the elbow and endpoint ([x_1, y_1], [x_2, y_2]) to completely describe the system state. We only need the elbow coordinates to distinguish between the two possible solutions for each prescribed endpoint (the "right-handed" and "left-handed" poses). Thus, if we restrict the system further by only allowing "right-handed" poses, the system state can be described simply with the endpoint coordinates. For this study, we chose to use the Euclidian coordinates of the manipulator's target endpoint (x^t,y^t) as the network inputs. Error was measured from the manipulator's endpoint $(x_2(t),y_2(t))$ from a target position (x^t,y^t) over time with the following performance criteria:

$$\text{Error} = \sum_{t_1}^{t_{max}} \left\{ [x_2(t) - x_t(t)]^2 + [y_2(t) - y_t(t)]^2 \right\}$$

where the interval from t_1 to t_{max} was chosen empirically to be a reasonable time to complete the movement.

6.4.3.2 Preliminary Results

Initially, we tried to train the network to produce the switching times as a function of several different target states, beginning from a given initial state ($[x_1,y_1,x_2,y_2] = [1,0,2,0]$ or $[\theta_1,\theta_2] = [0,0]$). Training on this problem was unsuccessful. The network would usually move the arm into a compromise position between all of the target states. Additional training could not get the system out of this apparent local minimum.

In a scaled-down problem, we attempted to drive the system in a *single* movement (from $[x_1,y_1,x_2,y_2] = [1,0,2,0]$ to $[x_1,y_1,x_2,y_2] = [0,1,0,2]$). The resultant arm trajectory is shown in Figure 6.14. Note that the plant had not come to a complete stop at the final position. This was to be expected since the system had no velocity damping.

The system is designed to seek out compromise solutions between time optimality and accuracy. This underscores the importance of how we choose to measure performance. This plant is

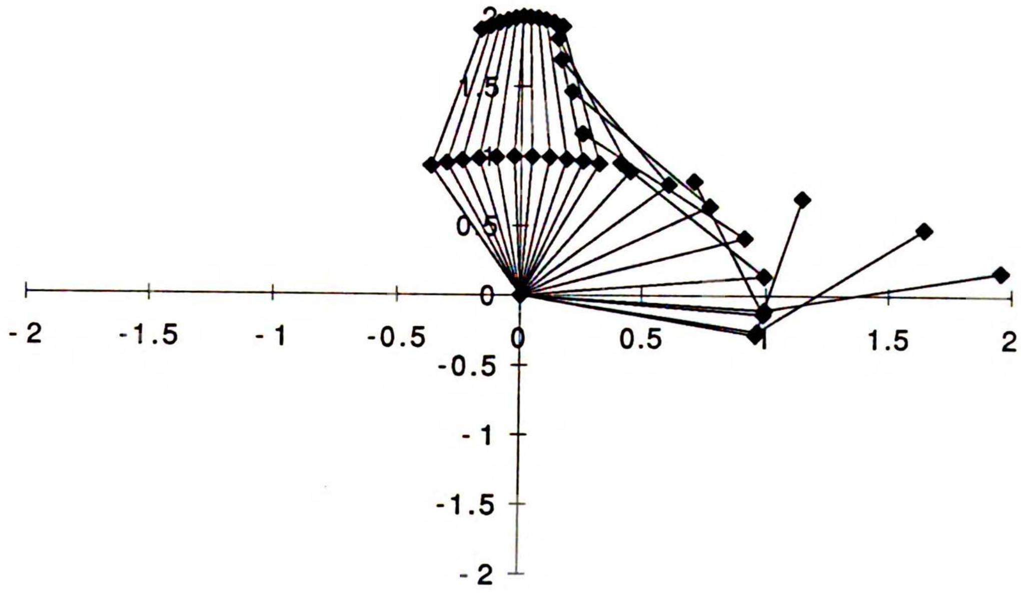


Figure 6.14: Dynamic trajectory Generated by a neural network

highly sensitive to variations in switching times. Perhaps control could be made simpler by adding some viscous damping to the plant. In addition, we have not studied the relative effectiveness of using θ_1,θ_2-space for system inputs or measuring performance in angular error. In the discussion, we discuss variations of this method for more complex plants which may avoid some of these difficulties in future studies.

6.5 DISCUSSION

Neural networks have been studied for solving problems involving classification, association and optimization. Recently, the mapping capabilities of neural networks have also been exploited to solve function approximation problems. In this chapter, we believe, we have demonstrated the use of neural networks to solve a function generation problem as well. Specifically, we have demonstrated a method of generating the optimal control function of a dynamical system by reformulating the control problem as an approximation problem and demonstrated it for simple examples.

6.5.1 FUTURE OUTLOOK

It seems reasonable that this method could be extended to several switching times. Alternative parameterization schemes may prove useful. In addition to switching times, a neural network could be applied to estimate the coefficients for a Fourier or Taylor series of the control signal. Other possibilities include parameterization of pulsed wavelets or Haar polynomials (Bremermann 1968, Andrews 1972). Our choice of using the switching times as the estimated signal parameters was a direct result of the chosen performance criterion (time-optimal control). In other cases, one may prefer to change the performance function being optimized, such as minimum *power* control. Current efforts on extending this methodology to more complex control tasks (such as multiple-link robotic arms); however, will require more extensive simulations and study. If successful, this could prove to be a powerful tool.

6.5.2 RELATION TO BIOLOGICAL CONTROL

The motivation behind the desire to address the above problem was rooted in neurobiological considerations. For example, it is well known that skilled motor movements, such as ordinary hand positioning movements, consist of "open-loop steps", or incremental "jumps" in position (Crossman 1959,1983; Anderson 1981). Control of saccadic eye movements is open-loop, and possibly, time-optimal (Lehman and Stark 1979, 1983). Taken as a whole, the movement of a skilled, or trained, machine operator is in some sense optimal (Inooka 1990). Natural neural networks obviously play a crucial role in generating these optimal control signals.

The signal generators in this study can be considered analogous to the Central Pattern Generators encountered in biological systems. High level control of movement is thought to involve the coordination or modulation of existing Central Pattern Generators (Selverston 1980). In this study, we used the chemotaxis algorithm to coordinate the "wiring diagram" of a network modulating a signal generator.

A biased random walk is the mathematical analog of a trial-and-error process. We have presented this "algorithm" as a biologically plausible learning rule and demonstrated it's effectiveness at training neural networks to perform non-trivial functions, such as Boolean mappings, pattern recognition and motor control. The same "trial-and-error" processes, then, may be involved in the refinement of cortical maps, the attainment of perceptual discrimination abilities, and the acquisition of skilled movement. We hope that it will be possible to find experimental evidence for the proposed learning rule.

REFERENCES

David H. Ackley, G.E. Hinton and Terrence J. Sejnowski, "A Learning Algorithm for Boltzmann Machines", *Cognitive Science*, Vol. 9, pp. 147-169 (1985).

David H. Ackley, *A Connectionist Machine for Hillclimbing*, Kluwer Academic Publishers, Boston (1987).

L B Almeida, "A Learning Rule for Asynchronous Perceptrons with Feedback in a Combinatorial Environment", *Proc. IEEE First Int'l Conf. Neural Networks* (1987).

Wolfgang Alt, "Biased random walk models for chemotaxis and related diffusion approximations", *J. Mathem. Biology*, Vol. 9, pp. 147-177 (1980).

T.J. Anastasio and David A. Robinson, "The Distributed Representation of Vestibulo-Oculomotor Signals by Brain-Stem Neurons", *Biol. Cybern.*, v 61:pp. 79-88 (1989).

J.R. Anderson (Ed.) *Cognitive Skills and Their Acquisition*, Lawrence Erlbaum Associates, Hillsdale, N.J. (1981).

Russell W. Anderson and V. Vemuri, "Neural Networks Can Be Used For Open-Loop, Dynamic Control", To appear in: *Intn'l J. Neural Networks: Research & Applications* (1991).

Harry C. Andrews, *Mathematical Techniques in Pattern Recognition*, Wiley-Interscience, New York (1972).

Chiye Aoki and Philip Siekevitz, "Plasticity in Brain Development", *Scientific American*, Vol. 259, No. 6, pp. 56-64 (December 1988).

Michael A. Arbib, *Brains, Machines, and Mathematics*, Second Edition, (First Edition: McGraw 1964), Springer-Verlag, New York (1987).

Scott Austin, "Genetic Solutions To XOR Problems", *AI Expert*, pp. 52-57 (December 1990).

Bill Baird, Ph.D. Thesis, U.C. Berkeley (1990).

Dana H. Ballard, "Modular Learning in Neural Networks", *AAAI National Conference on Artificial Intelligence,* pp. 279-284 (1987).

Robert L. Barron, "Self-Organizing and Learning Control Systems", in: *Cybernetic Problems in Bionics* (Bionics Symposium, May 2-5, 1966, Dayton, Ohio), New Yor, Gordon and Breach, pp. 147-203 (1968).

Robert L. Barron, "Adaptive Flight Control Systems", in: *Principles and Practice of Bionics* (NATO AGARD Bionics Symposium, Brussels, Belgium, Sept. 18-20, 1968), pp. 119-167 (October 1970).

Andrew G Barto, Richard S Sutton and Peter S Brouwer, "Associative Search Network: A Reinforcement Learning Associative Memory", *Biological Cybernetics*, Vol. 40, pp. 201-211 (1981).

Andrew G Barto and Richard S Sutton, "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems", *IEEE Transactions on Systems, Man, and Cybernetics*", Vol. SMC-13, No. 5, pp. 835-846 (1983).

Jacob D. Bekenstein and Marcello Schiffer, "Quantum Limitations on the Storage and Transmission of Information", *International Journal of Modern Physics C* (in press 1990).

R. Bellman, I. Glicksberg and O. Gross, "On the 'Bang-Bang' Control Problem", *Quarterly of Applied Mathematics*, 14: 11-18 (1956) Reprinted in *Optimal and Self-Optimizing Control*, ed. R. Oldenburger. M.I.T. Press, Cambridge, Mass. (1966).

T.J. Bennett, and J.R. Murphy, "Analysis of seismic discrimination using regional data from western United States events", *Bull. Seism. Soc. Am.*, Vol. 76, pp. 1069-86 (1986).

Howard Berg, "How Bacteria Swim", *Scientific American*, Vol. 233(2), pp. 36-44 (1975).

Howard Berg, *Random Walks in Biology*, Princeton University Press, Princeton (1983).

P.J. Bergold, J.D. Sweatt, I. Winicov, K.R. Weiss, E.R. Kandel and J.H. Schwartz, "Protein Synthesis During Acquisition of Long-Term Facilitation is Needed for the Persistent Loss of Regulatory Subunits of the *Aplysia* cAMP-Dependent Protein Kinase", *PNAS (USA)*, Vol. 87, No. 10, PP. 3788-91 (May 1990).

M. Bertero, T.A. Poggio and V. Torre, "Ill-Posed Problems in Early Vision", *Proc. IEEE*, Vol. 76 (8): pp. 869-887 (August 1988).

W.W. Bledsoe, "The Use of Biological Concepts in the Analytical Study of Systems", Technical Report, Panoramic Research Inc., Palo Alto, CA (1961a).

W.W. Bledsoe, "Lethally Dependent Genes Using Instant Selection", Technical Report, Panoramic Research Inc., Palo Alto, CA (1961b).

W.W. Bledsoe, "A Quantum-Theoretical Limitation of the Speed of Digital Computers", *IRE Trans. Elec. Comp.*, Vol. EC-10, No. 3 (Sept., 1961c).

T. Boseniuk, W. Ebeling and A. Engel, "Boltzmann and Darwin Strategies in Complex Optimization", *Physics Letters A*, Vol. 125, pp. 307-310 (1987).

Hans J. Bremermann, "The Evolution of Intelligence", ONR Technical Report No. 1, Contract Nonr 477(17), University of Washington, Seattle (1958).

Hans J. Bremermann, "Optimization Through Evolution and Recombination", in: *Self-Organizing Systems*, Yovits, Jacobi and Goldstein, eds., Spartan Books, Washington, D.C. (1962).

Hans J. Bremermann and M. Rogson, "An Evolution-Type Search Method for Convex Sets", Technical Report, Contracts Nonr 222(85) and 3656(08), Berkeley, CA (1964).

Hans J. Bremermann, M. Rogson and S. Salaff, "Global Properties of Evolution Processes", In: *Natural Automata and Useful Simulations*, H.H. Pattee, E.A. Edelsack, Louis Fein and A.B. Callahan, eds., Spartan Books, Washington, D.C., pp. 3-41 (1966).

Hans J. Bremermann, "Pattern Recognition, Functionals, and Entropy", *IEEE Transactions on Biomedical Engineering*, Vol. BME-15 (3), pp. 201-207 (July 1968).

Hans J. Bremermann, "A Method of Unconstrained Global Optimization", *Mathematical Biosciences*, Vol. 9, pp. 1-15 (1970).

Hans J. Bremermann, "Chemotaxis and Optimization", *J. of the Franklin Institute*, (Special Issue: Mathematical Models of Biological Systems) Vol. 297, pp. 397-404 (1974a).

Hans J. Bremermann, "Complexity of Automata, Brains, and Behavior", *Lecture Notes in Biomathematics, Vol. 4: Physics and Mathematics of the Nervous System,* S. Levin (ed.), Springer-Verlag, Berlin, pp. 304-331 (1974b).

Hans J. Bremermann and Russell W. Anderson, "An Alternative to Back-propagation: A Simple Rule of Synaptic Modification For Neural Net Training and Memory", Technical Report: U.C.Berkeley Center for Pure and Applied Mathematics PAM-483 (1990).

Hans J. Bremermann and Russell W. Anderson, "How the Brain Adjusts Synapses - Maybe", To appear in: *Festschrift for Woody Bledsoe*, R.S. Boyer (ed.), Chapter 6, pp. 119-147, Kluwer Academic Pub. (1991).

Felix E. Browder (Ed.), *Mathematical Developments Arising from Hilbert Problems*, American Mathematical Society, Providence, RI (1976).

D.V. Buonomano, and J.H. Bryne, "Long-Term Synaptic Changes Produced by a Cellular Analog of Classical Conditioning in *Aplysia*", *Science*, Vol. 249, pp. 420-3 (27 July 1990).

David Ceperley and Berni Alder, "Quantum Monte Carlo", *Science*, Vol. 231, pp. 555-560 (7 Feb 1986).

D.E. Chavez, and K.F. Priestley, "Measurement of frequency dependent Lg attenuation in Great Basin", *Geophys. Res. Lett.*, Vol. 13, pp. 551-554 (1986).

C.H. Chen, "On the Relationships Between Statistical Pattern Recognition and Artificial Neural Networks", *1990 IEEE SMC Conf.*, Los Angeles (Nov. 1990).

Fu-Chuang Chen, "Back-Propagation Neural Networks for Nonlinear Self-Tuning Adaptive Control", *IEEE Control Systems Mag.*, Vol. 10(3): 44-48 (April 1990).

Daniel L. Chester, "Why Two Hidden Layers are Better than One", *Proc. IJCNN*, Washington, D.C., Lawrence Erlbaum Assoc., Vol. I, pp. 265-68 (Jan 15-19, 1990a).

Daniel L. Chester, "A Comparison of some Neural Network Models of Classical Conditioning", Proc. 5th IEEE Int'l Symposium on Intelligent Control, Philadelphia, PA, Vol. 2, pp. 1163-1168 (Sept. 5-7, 1990b).

Michael Conrad, *Adaptability*, (Chapter 10), Plenum Press, N.Y. (1983).

Michael Conrad, "The brain-machine disanalogy", *Biosystems*, Vol. 22: 197-213 (1989).

J.D. Cowan, "Discussion: McCulloch-Pitts and Related Neural Nets From 1943-1989", *Bull. Mathematical Biology*, Vol. 52, No1/2, pp. 73-97 (1990).

Francis Crick, "The Recent Excitement about Neural Networks", *Nature*, Vol. 337, pp. 129-132 (12 January 1989).

Francis Crick, "Neural Edelmanism", *Trends in Neurosciences*, Vol. 12 (7):240-248 (1989)

E.R.F.W. Crossman, "A Theory of the Acquisition of Speed-Skill", Ergonomics, Vol.2 (2):153-166 (Feb. 1959).

E.R.F.W. Crossman and P.J. Goodeye, "Feedback Control of Hand-Movement and Fitt's Law", Quarterly Journal of Experimental Psychology, Vol. 35A:251-278 (1983).

Adele Cutler, "Optimization Methods in Statistics", Ph.D Thesis, Department of Statistics, University of California, Berkeley (1988).

G. Cybenko, "Approximation by Superpositions of a Sigmoidal Function", *Math. Contr., Signal & Sys.*, Vol. 2, pp. 303-14 (1989).

Shawn P. Day and Daniel S. Camporese, "A Stochastic Training Technique for Feed-Forward Neural Networks", *Proc. IEEE Conf. Neural Networks*, June 17-21 (1990).

O. Dahlman and H. Israelson, *Monitoring Underground Nuclear Explosions*, Elsevier Scientific Publishing co., Amsterdam (1977).

Kurt Deno, Ph.D. Thesis, U.C. Berkeley, Dept. EECS (1991).

Farid U. Dowla, A.J. DeGroot, S.R. Parker and V. Vemuri, "Back-propagation neural networks: systolic implementation for seismic signal filtering", *Intn'l J.Neural Networks*, Vol. 1 (3), pp. 138-153 (July 1989).

Farid U. Dowla, Steven R Taylor and Russell W. Anderson, "Seismic Discrimination with Artificial Neural Networks: Preliminary Results with Regional Spectral Data", *Bulletin of the Seismological Society of America*, Vol. 80, No. 5, pp. 1346-1373 (October 1990).

Kenji Doya and Shuji Yoshizawa, "Memorizing Oscillatory Patterns in the Analog Neuron Network", *Internat. Joint Conf. Neural Networks*, pp. I-27-32, Washington, D.C. (1989).

R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York (1973).

S.A. Eaton and T.E. Salt, "Modulatory Effects of Serotonin on Excitatory Amino Acid Responses and Sensory Synaptic Transmission in the Ventrobasal Thalamus", Neuroscience, Vol. 33, No. 2, pp. 285-92 (1989).

W. Ebeling, A. Engel, B. Esser and R. Feistel, "Diffusion and Reaction in Random Media and Models of Evolution Processes", *J. Statistical Physics*, Vol. 37, No. 3/4, pp. 369-384 (1984).

G.M. Edelman, *Neural Darwinism*, Basic Books, New York (1987).

Manfred Eigen, J. McCasgill and Peter Schuster, "Dynamics of Darwinian Molecular Systems", *J. phys. Chem*, (In press).

Manfred Eigen, "Macromolecular Evolution: Dynamical Ordering in Sequence Space", In: *Emerging Synthesis in Science*, D. Pines, ed., Addison-Wesley, Redwood City, CA, pp. 21-42 (1988).

Jeffrey L Elman, "Finding Structure in Time", Center for Research in Language Tech. Report #8801, La Jolla: University of California, San Diego (1988).

Jerome A. Feldman, "A connectionist model of visual memory", In: *Parallel models of associative memory*, G.E. Hinton and J.A. Anderson, eds., Erlbaum, Hillsdale, N.J., pp. 49-81 (1981).

Jerome A. Feldman and Dana H. Ballard, "Connectionist Models and their Properties", *Cognitive Science*, Vol. 6, pp. 205-254 (1982).

N.A. Flavahan and P.M. Vanhoutte, "Threshold Phenomena and Interactions Between Receptors", J. of Cardiovascular Pharmacology, Vol. 11 suppl. 1, pp. S67-72 (1988).

Walter Fontana, W. Schnabl and Peter Schuster, "Physical Aspects of Evolutionary Optimization and Adaptation", *Phys. Rev. A*, Vol. 40, pp. 3301-21 (1989).

Walter J. Freeman, "The Physiology of Perception", *Scientific American*, Vol. 264(2): 78-85 (February 1991).

Ken-ichi Funahashi, "On the approximate realization of continuous mappings by neural networks", *Neural Networks*, Vol. 2(3): 183-192 (1989).

S. Geman and D. Geman, "Stochastic relaxation, Gibbs distribution, and Bayesian restoration of images", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 6, pp. 721-741 (1984); also in: *Neurocomputing: Foundations of Research*, James A. Anderson and Edward Rosenfeld, eds., MIT Press, Cambridge, MA (1988).

David L. Glanzman, Eric R. Kandel and Samuel Schacher, "Target-Dependent Structural Changes Accompanying Long-Term Synaptic Facilitation in Aplysia Neurons", *Science*, Vol. 249, pp. 799-802 (17 August 1990).

R.E. Glaser, S.R. Taylor, M.V. Denny and E.S. Vergino, "Regional discriminants of NTS explosions and western U.S. earthquakes: multivariate discriminants", Tech. Report No. UCID-20930, Lawrence Livermore National Laboratory.

R. Paul Gorman and Terrence J. Sejnowski, "Learned Classification of Sonar Targets Using a Massively Parallel Network", *IEEE ASSP*, Vol. 36 (7):1135-1140 (1988).

Kamil A. Grajski and Michael M. Merzenich, "Hebb-Type Dynamics is Sufficient to Account for the Inverse Magnification Rule in Cortical Somatotopy", *Neural Computation* (1990).

Stephen Grossberg, In: *Neurocomputing: Foundations of Research*, James A. Anderson and Rosenfeld, eds., MIT Press, Cambridge (paper number 24) (1988).

Stephen Grossberg and Nestor A Schmajuk, "Neural dynamics of adaptive timing and temporal discrimination during associative learning", *Neural Networks* , vol. 2(2): 79-102 (1989).

H. Haken, *Neural and Synergetic Computers*, Springer-Verlag, Berlin Heidelberg (1988).

John B. Hampshire and Barak A. Pearlmutter, "Equivalence Proofs for Multilayer perceptron Classifiers and the Bayesian Discriminant Function", *Proc. 1990 Connectionist Models Summer School*, Elman, Sejnowski and Hinton (Eds.), Morgan Kaufmann, San Mateo, CA (1990).

E.J. Hartman, J.D. Keeler and J.M. Kowalski, "Layered Neural Networks with Gaussian Hidden Units as Universal Approximations", *Neural Computation*, Vol. 3 (2), pp. 210-215 (1990).

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

Robert Hecht-Nielsen, "Kolmogorov's Mapping Neural Network Existing Theorem", *Proc. Intl. Conf. on Neural Networks*, New York, Vol. III, pp. 11-14 (1987).

Robert Hecht-Nielsen, *Neurocomputing*, Addison-Wesley (1990).

G.E. Hinton and J.L. McClelland, "Learning representations by recirculation", In: *Neural Information Processing Systems*, D.Z. Anderson, ed., American Institute of Physics, New York (1988).

G.E. Hinton, "Connectionist Learning Procedures", *Artificial Intelligence*, Vol. 40, No. 1, pp. 143-150 (1989).

K.U. Hoffgen and H.P. Siemon, "Approximation of Functions with Feedforward nets", Dept. of Comp. Sci., Univ. of Dortmund, D-4600 Dortmund, P.O.Box 500 500, FRG, Report No. 346 (April 1990).

John H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor (1975).

John J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons", *PNAS (USA)*, Vol. 81, pp. 3088-92 (1984).

John J. Hopfield and David W. Tank, "'Neural' Computation of Decisions in Optimization Problems", *Biological Cybernetics*, Vol. 52, pp. 141 (1985).

John J. Hopfield and David W. Tank, "Computing with Neural Circuits: A Model, *Science*, pp. 625-633 (8 Aug 1986).

Berthold K.P. Horn, "Kinematics of the MIT-AI-Vicarm Manipulator", *MIT AI Lab. Working Paper 69* (1974).

Shih-Chi Huang and Yih-Fang Huang, "Bounds on the Number of Hidden Neurons in Multilayer Perceptrons", *IEEE Trans. on Neural Networks*, Vol. 2, No. 1, pp. 47-55 (Jan. 1991).

David Hubel and T.N. Wiesel, "Receptive Fields, Binocular and Functional Architecture in the Cat's Visual Cortex", *J. Physiol.*, Vol. 160, pp. 106-154 (1962).

David Hubel, *Eye, Brain and Vision*, Scientific American Library Series #22. (1988).

H. Inooka and Koitabashi, T., "Experimental Studies of Manual Optimization in Control Tasks", *IEEE Control Systems Magazine*, pp 20-23 (August 1990).

M.D. Intriligator, *Mathematical Optimization and Economic Theory*, Prentice Hall, Inc., Englewood Cliffs, N.J. (1971).

H. Jack, D.M.A. Lee, R.O. Buchal and W.H. ElMaraghy, "Evaluations of Neural networks for Robot Inverse Kinematics", Submitted to: *IEEE Int'l Conf. on Robotics and Automation*, Cincinnati, Ohio (May 1990).

Eric M. Johansson, Farid U. Dowla and D.M. Goodman, "Backpropagation Learning for Multi-Layer Feed-Forward Neural Networks Using the Conjugate Gradient Method", submitted to *IEEE Transactions on Neural Networks*, UCRL-JC-1850, Lawrence Livermore National Laboratory (September 26, 1990).

M.I. Jordan, "Attractor dynamics and parallelism in a connectionist sequential machine", *Proc. Eighth Annual Conf. of the Cognitive Science Soc.*, pp. 531-546 (1987).

Charles C. Jorgensen, "Development of a Sensor Coordinated Kinematic Model for neural Network Controller Training", Research Institute for Advanced Computer Science (RIACS), NASA Ames Research Center Tech. Report RAICS 90.28 (August 1990a).

Charles C. Jorgensen, "Distributed Memory Approaches for Robotic Neural Controllers", Research Institute for Advanced Computer Science (RIACS), NASA Ames Research Center Tech. Report RAICS 90.29 (August 1990b).

M.E. Kahn, "The Near-Minimum-Time Control of Open-Loop Articulated Kinematic Chains", Stanford Artificial Intelligence Memo 106 (1969).

J.A. Kauer, R.C. Malenka and R.A. Nicoll, *Nature*, Vol. 334, pp. 249-252 (1988).

Stuart A. Kauffman and S. Levin, "Towards a General Theory of Adaptive Walks on Rugged Landscapes", *J. Theoret. Biol.*, Vol. 128, pp. 11-45 (1987).

Evelyn Fox Keller and Lee Segel, *J. of Theoretical Biology*, Vol. 26, pp. 399 (1970).

Mary B. Kennedy, "Synaptic Memory Molecules", *Nature*, Vol. 335, pp. 770-772 (27 Oct 1988).

A. Harry Klopf, "Classical conditioning phenomena predicted by a drive-reinforcement model of neuronal function", In: *Neural Models of Plasticity: Experimental and Theoretical Approaches*, J.H. Byrne and W.O. Berry (eds.), Chapter 7, pp. 104-132, Academic Press (1989).

T. Kohonen, *Self-Organization and Associative Memories*, Springer-Verlag, Berlin (1984).

G.G. Kolmogorov, *Dokl. Akad. Nauk. SSR*, Vol. 114, pp. 953-956 (1957).

G.G. Kolmogorov, "On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition", *American Mathematical Society Translations*, Series 2, Vol. 28, pp. 55-59.

Daniel Koshland, *Bacterial chemotaxis as a model behavioral system*, Raven Press, New York (1980).

P.A. Lachenbruch and R.M. Mickey, "Estimation of error rates in discriminant analysis", *Technometrics*, Vol. 10, pp. 1-11 (1968).

Douglas M. Lee, H. Jack, W.H. ElMaraghy and R.O. Buchal, "Neural Network Solutions for Robot Inverse Kinematic Control", Tech. Report DAMRL 90-07-01, Submitted to: *Proc. of the CSME mechanical Eng. Forum*, Toronto, Ontario (Jan 1990).

S.R. Lehky and Terrence J. Sejnowski, "Computing 3-D Curvatures from Images of Surfaces Using a Neural Model, *Nature*, Vol. 333, pp. 452 (1988).

S.R. Lehky and Terrence J. Sejnowski, "Neuronal Model of Stereoacuity and Depth Interpolation Based on a Distributed Representation of Stereo Disparity", *Journal of Neuroscience*, Vol. 10, No. 7, pp. 2281-2299 (July 1990).

Steven Lehman and L. Stark, "Simulation of Linear and Nonlinear Eye movement Models: Sensitivity Analyses and Enumeration Studies of Time Optimal Control", *J. of Cybernetics and Information Science*, v 4:211-43 (1979).

Steven Lehman and Lawrence W. Stark, "Multipulse Controller Signals. II. Time Optimality", *Biological Cybernetics"*, v47:234-237 (1983).

George Leitmann,*The Calculus of Variations and Optimal Control*, Plenum Press, New York, NY. (1981).

Richard P. Lippman, "Introduction to computing with neural nets", *IEEE ASSP magazine*, pp. 4-22 (1987).

Richard P. Lippmann, "Review of Neural Networks for Speech Recognition", *Neural Computation*, MIT Press, Vol. 1(1): 1-38 (1989).

Shawn R. Lockery, G. Wittenberg, W.B. Kristan and G.W. Cottrell, "Function of Identified Interneurons in the Leech Elucidated Using Neural Networks Trained by Back-Propagation", *Nature*, Vol. 340, pp. 468-71 (10 August 1989).

Shawn R. Lockery, Y. Fang and T.J. Sejnowski, "A Dynamical Neural Network Model of Sensorimotor Transformations in the Leech", *Intn'l Joint Conf. Neural Networks*, I-183-188 (June 1990).

G.G. Lorentz, "The 13th Problem of Hilbert", in: *Mathematical Developments Arising from Hilbert Problems*, F.E. Browder (Ed.), American Mathematical Society, Providence, RI (1976).

Gary Lynch, *Synapses, Circuits, and the Beginnings of Memory*, Bradford/MIT Press, Cambridge, MA (1986).

Catherine A Macken and Alan S. Perelson, "Protein Evolution on Rugged Landscapes", *PNAS(USA)*, Vol. 86, pp. 6191-5 (August 1989).

Catherine A Macken, Patrick S. Hagan and Alan S. Perelson, "Evolutionary Walks on Rugged Landscapes", *SIAM J. Appl. Math.* (in press:1991).

S.F. Masri, G.A. Bekey and F.B. Stafford, "A Global Optimization Algorithm Using Adaptive Random Search", *Applied Mathematics and Computation,* v 7:353-375 (1980).

J.L. McClelland and D.E. Rumelhart, "An interactive activation model of context effects in letter perception. Part 1", *Psychological Review*, Vol. 88, pp. 375-407 (1981).

W.S. McCulloch and W. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity", *Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115-133 (1943).

Bartlett W. Mel, *Connectionist Robot Motion Planning*, Academic Press, Boston, San Diego (1990).

M.M. Merzenich, G. Recanzone, W.M. Jenkins, T.T. Allard and R.J. Nudo, "Cortical Representational Plasticity", pp. 41-67, (In: Rakic and Singer 1988).

M.M. Merzenich, R.J. Nelson, J.H. Kaas, M.P. Stryker, W.M. Jenkins, J.M. Zook, M.S. Cynader and A. Schoppman, "Variability in Hand Surface Representations in Areas 3b and 1 in Adult Owl and Squirrel Monkeys", *J. of Comparative Neurology,* Vol. 258(2), pp. 281-96 (April 8, 1987).

N. Metropolis and S. Ulam, "The Monte Carlo Method", J. *Amer. Statistical Association*, Vol. 44 No. 247, pp. 335-341 (1949).

N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, "Equation of state calculations for fast computing machines", *J. of Chemical Physics*, Vol. 21, pp. 1087-1092 (1953).

J.P. Miller and A.I. Selverston, "Mechanisms underlying pattern generation in lobster stomatogastric ganglion as determined by selective inactivation of identified neurons IV. Network properties of the pvloric system", *J. Neurophysiol.*, Vol. 48, pp. 1416-1432 (1982).

Jaime Milstein, "Estimation of the dynamical parameters of the Calvin photosynthesis cycle, optimization, and ill conditioned inverse problems." (LBL document 4264) Ph.D. Thesis, U.C. Berkeley (1975).

Marvin Minsky, "Steps Toward Artificial Intelligence", *Proc. IRE,* Vol. 49, pp. 8-30 (Jan 1961).

Marvin Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*, M.I.T. Press, Cambridge, Mass (1969).

D. Montana and L. Davis, "Training Feedforward Neural Networks Using Genetic Algorithms", *Proc. 11th IJCAI* (1989).

P.G. Montarolo, E.R. Kandel and S. Schacher, "Long-Term Heterosynaptic Inhibition in *Aplysia*", *Nature*, Vol.333, pp. 171-4 (1988).

Zoltan Molnar and Colin Blakemore, "Lack of regional specificity for connections formed between thalamus and cortex in coculture", *Nature*, Vol. 351, pp. 475-477 (June 1991).

Anthony N. Mucciardi, "Neuromine Nets as the Basis for the Predictive Component of Robot Brains", in: *Cybernetics, Artificial Intelligence, and Ecology*, Herbert W. Robinson and Douglas E. Knight (eds.), (Fourth Annual Symposium Amer. Soc. of Cybernetics), Spartan Books, pp. 159-193 (1972).

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

Ralph Nossal, "Mathematical Theories of Topotaxis", *Lecture Notes in Biomathematics*, Vol. 38, Springer-Verlag, pp. 410-439

A. Okubo, "Diffusion and Ecological Problems: Mathematical Models", *Biomathematics*, Vol. 10, Springer Verlag, Berlin Heidelberg New York (1980).

John C. Pearson, Leif H. Finkel and Gerald M. Edelman, "Plastiscity in the Organization of Adult Cerebral Cortical Maps: A Computer Simulation Based on Neuronal Group Selection", *J. of Neuroscience*, Vol. 7(12): 4209-4223 (Dec. 1987).

Alan S. Perelson and Stuart A. Kauffman, *Molecular Evolution an Rugged Landscapes: Proteins, RNA and the Immune System*, Vol. IX, Addison-Wesley, Redwood City, CA (1991).

C. Peterson and J.R. Anderson, "A Mean Field Theory Learning Algorithm for Neural Networks", MCC Tech. Rept. E1-259-87, Microelectronics and Computer Technology Corporation, Austin, TX (1987).

Fernando J. Pineda, "Generalization of backpropagation to recurrent and higher order neural networks", *Physics Review Letters*, pp. 602-611 (1988).

Tomasio Poggio and Chistoph Koch, "Synapses That Compute Motion", *Scientific American*, vol. 256(5): 46-52 (May 1987).

T. Poggio and S. Edelman, "A network that learns to recognize three-dimensional objects", *Nature*, Vol. 343: 263-266 (18 Jan. 1990).

T. Poggio and F. Girosi, "Regularization Algorithms for Learning That are Equivalent to Multilayer Neural Networks", *Science*, Vol. 247, pp. 978-82 (23 February 1990).

P.W. Pomeroy, W.J. Best and T.V. McEvilly, "Test ban treaty verification with regional data - a review", *Bull. Seismological Soc. of America*, Vol. 72, Part B, S89-S129 (1982).

L.S. Pontryagin, V.G. Boltyanskii, R.V. Gamkrelidze and E.F. Mischenko, *The Mathematical Theory of Optimal Processes*, trans. by K.N. Trirogoff. Interscience Publishers, John Wiley & Sons (1962).

D. Psaltis, A. Sideris and A.A. Yamamura, "A Multilayered Neural Network Controller", *IEEE Control Sys. Mag.*, pp.17-21 (April 1988).

Ning Qian and Terrence J. Sejnowski, "Predicting The Secondary Structure of Globular Proteins Using Neural Network Models", *J. Molec. Biol.*, Vol. 202, pp. 865-884 (1988).

P. Rakic and W. Singer, *Neurobiology of Neocortex*, (Dahlem Conferences Report No. 42) Wiley-Interscience, New York (1988).

Anna W. Roe, Sarah L. Pallas, Jong-On Hahm and Mriganka Sur, "A Map of Visual Space Induced in Primary Auditory Cortex", *Science*, Vol. 250, pp. 818-20 (9 November 1990).

Frank Rosenblatt, "The Perceptron, a probabilistic model for information storage and organization in the brain", *Psych. Rev.*, Vol. 62, pp. 386-408 (1958).

Frank Rosenblatt, *Principles of Neurodynamics*, Spartan Books, Washington, D.C. (1962).

David E. Rumelhart, Geoffrey E. Hinton and R.J. Williams, "Learning Internal Representations by Error Propagation, In: *Parallel Distributed Processing Vol.1*, D.E. Rumelhart and J.L. McClelland, eds., MIT Press, Cambridge, MA pp. 318-362 (1986).

D.E. Rumelhart and J.L. McClelland, "An interactive activation model of context effects in letter perception. Part 2", *Psychological Review*, Vol. 89, pp. 60-94 (1982).

D.E. Rumelhart and J.L. McClelland, eds., *Parallel Distributed Processing Vol.1*, MIT Press, Cambridge, MA (1986).

D.W. Ruck, S.K. Rogers, M. Kabrinsky, M.E. Oxley and B.W. Suter, "The Multilayer Perceptron as an Approximation to a Bayes Optimal Discriminant Function", *IEEE Trans. on Neural Networks*, Vol. 1, No. 4, pp.296-8 (Dec. 1990).

Peter Schuster and K. Sigmund, "Dynamics of Evolutionary Optimization", *Ber. Bunsenges. phys. Chem.*, Vol. 89, pp. 668-682 (1985).

Peter Schuster and Jorg Swetina, "Stationary Mutant Distributions and Evolutionary Optimization", *Bulletin of Mathematical Biology*, Vol. 50, No. 6, pp. 635-660 (1988).

David R. Seidl and Robert D. Lorenz, "Summary of a Structure by which a Recurrent Neural Network Can Approximate a Nonlinear Dynamic System", *Proc. Intn'l AMSE Conf. Neural Networks*, San Diego, CA (May 29-31, 1991).

Terrence J. Sejnowski and Charles R. Rosenberg, "Parallel Networks that Learn to Pronounce English Text", *Complex Systems*, Vol. 1, pp. 145-168 (1987).

Allen I. Selverston, "Are Central Pattern Generators Understandable?", *Behavioral and Brain Sciences*, vol. 3, pp. 535-571 (1980).

Yu. A. Shreider, *The Monte Carlo Method*, Pergamon Press, Oxford, Pure and Applied Mathematics, Vol. 87, Translation from Russian (1966).

Hava Seigelman and Eduardo Sontag, "Neural Nets are Universal Computing Devices", Technical Report SYCON-91-08, Rugters University, Center for Systems and Control (May 1991).

Christine A. Skarda and Walter J. Freeman, "How Brains Make Chaos in Order to Make Sense of the World", *Behav. and Brain Sciences*, Vol. 10(2): 161-195 (1987).

Robert Smalz and Michael Conrad, "A Credit Apportionment Algorithm for Evolutionary Learning with Neural Networks", (preprint), Wayne State University, Detroit (1990).

Patric K. Stanton and Terrence J. Sejnowski, "Associative long-term depression in the hippocampus induced by hebbian covariance", *Nature*, Vol. 339, pp. 215-218 (18 May 1989).

Charles F. Stevens, "Strengthening the synapses", *Nature*, Vol. 338, pp. 460-461 (6 April 1989).

Lawrence D. Stone, *Theory of Optimal Search,* Academic Press, New York, N.Y. (1975).

Gilbert Strang, *Linear Algebra and Its Applications* (Second Edition), Academic Press, New York (1980).

M.P. Stryker, J. Allman, C. Blakemore, J.M. Greuel, J.H. Kaas, M.M. Merzenich, P. Rakic, W. Singer, G.S. Stent, H. van der Loos and T.N. Wiesel, "Group Report: Principles of Cortical Self-Organization", pp. 115-136, (In Rakic and Singer 1988).

Gita Subba-Rao, "Calculation of the Minimum Energy Conformation of Biomolecules by Using a Global Optimization Technique. V. Preferred Conformations of the Thyrotropin-Releasing Hormone", *International J. of Quantum Chemistry*, Vol. 29, pp. 1177-1180 (1986).

Richard S. Sutton and Andrew G. Barto, "Toward a Modern Theory of Adaptive Networks: Expectation and Prediction", *Phychological Review*, Vol. 88 (2):135-170 (1981).

Roman R. Swiniarski, "Neural Recurrent State Estimator of Dynamic Systems. Neural Kalman Filtering", *Proc. Intn'l AMSE Conf. Neural Networks*, San Diego, CA (May 29-31, 1991).

Roman R. Swiniarski and J. Nieplocha, "Neural Estimation fro a Class of Nonlinear Systems", *Proc. Intn'l AMSE Conf. Neural Networks*, San Diego, CA (May 29-31, 1991).

Y. Takahashi, M.J. Rabins, and D.M. Auslander, *Control and Dynamic Systems*, Addison-Wesley, Reading, Mass. (1972).

Steven R. Taylor, N.W. Sherman and M.D. Denny, "Spectral Discrimination Between NTS Explosions and Western United States Earthquakes at Regional Distances", *Bull. Seismological Soc. of America*, Vol. 78, No. 4, 1563-79 (August 1988).

Steven R. Taylor, M.D. Denny, E.S. Vergino and M.V. Denny, "Spectral Discrimination between NTS explosions and western United States earthquakes", *Bull. Seismological Soc. of America*, Vol. 79, pp. 1142-1176 (1989).

Steven R. Taylor, "Discrimination of Nuclear Explosions and Earthquakes", *Energy and Technology Review* (LLNL, MS L-3, P.O.Box 808, Livermore, Ca, 94551) pp. 28-37 (March 1990).

Steven R. Taylor, "Discriminating between Nuclear Explosions and Earthquakes", *Energy and Technology Review* (LLNL, MS L-3, P.O. Box 808, Livermore, Ca, 94551) pp. 56-57 (July-August 1990).

Gerald Tesauro and Terrence J. Sejnowski, "A Parallel Network that Learns to Play Backgammon", *Artificial Intelligence*, Vol. 39, No. 3, pp. 357-390 (July 1989).

Gerald Tesauro and Bob Janssens, "Scaling Relationships in Back-propagation learning", *Complex Systems*, Vol. 2, pp. 39-44 (1988).

D. Tjøstheim, "Multidimensional discrimination techniques - theory and application", *Identification of Seismic Sources - Earthquake or Underground Explosion*, Husebye and Mykkeltveit (eds.), D. Reidel Publishing Co., pp. 663-694 (1981).

Fu-Sheng Tsung and Garrison W Cottrell, "A Sequential Adder Using Recurrent Networks", *Internat. Joint Conf. Neural Networks*, pp. II-133-39, Washington, D.C. (1989).

Alan M. Turing, "On computable numbers, with an application to the Entscheidungsproblem", *Proc. London Math. Soc.*, Vol. 45: 161-228 (1937).

J.J. Uicker, "Dynamic Force Analysis of Spatial Linkages", *Transactions ASME* (1967).

C. Van den Broeck and R. Kawai, "Learning in Feedforward Boolean Networks", *Physical Rev. A*, v. 42 (10): 6210-6218 (1990).

C. Van den Broeck and R. Kawai, "Generalization in Feedforward Neural and Boolean Networks", *Proc. AMSE Intn'l Conf. "Neural Networks"*, San Diego (May 29-31, 1990).

D.C. Van Essen , In: *Cerebral Cortex Vol. 3*, A. Peters and F.G. Jones, eds., Plenum Press, New York, pp. 259-324 (1985).

Waibel, *Neural Computation,* Vol. 1(1) (1989).

Eric A. Wan, "Neural Network Classification: A Baysean Interpretation", *IEEE Trans. on Neural Networks*, Vol. 1, No. 4 (Dec. 1990).

H. White, "Learning in Artificial Neural Networks: A Statistical Perspective", *Neural Computation*, Vol. 1, pp. 425-464 (1989).

Bernard Widrow and M.E. Hoff, "Adaptive Switching Circuits", *Institute of Radio Engineers, Western Electronic Show and Convention Record Part 4*, pp. 96-104 (1960).

J H Williams, M L Errington, M A Lynch and T V P Bliss, "Arachidonic Acid Induces a Long-Term Activity-Dependent Enhancement of Synaptic Transmission in the Hippocampus, *Nature*, Vol. 341, pp. 739-42 (26 October 1989).

R J Williams and David A. Zipser, "A Learning Algorithm foe Continually Running Fully Recurrent Neural Networks", Tech. Report #ICS-8805, University of California, San Diego (1988).

James M. Wilson, "Back-Propagation Neural Networks: A Comparison of Selected Algorithms and Methods of Improving Performance", *Proc. 2nd Annual Workshop Neural Networks WNN-AIND*, Auburn, Alabama (Feb. 11-13, 1991).

Lofti A. Zadeh, "Communication Fuzzy Algorithms", *Information and Control*, Vol. 12, p. 94-102 (1968).

4 5799