

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

U·M·I

University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600



Order Number 1346419

**Characterization of neural network simulations for optimal
classification of intraoperative electroencephalograph data**

Narus, Scott Patrick, M.S.

The University of Arizona, 1991

U·M·I

**300 N. Zeeb Rd.
Ann Arbor, MI 48106**

CHARACTERIZATION OF NEURAL NETWORK SIMULATIONS FOR
OPTIMAL CLASSIFICATION OF
INTRAOPERATIVE ELECTROENCEPHALOGRAPH DATA

by

Scott Patrick Narus

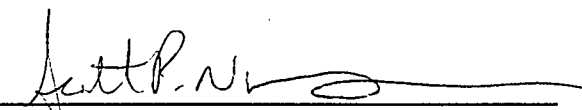
A Thesis Submitted to the Faculty of the
ELECTRICAL AND COMPUTER ENGINEERING DEPARTMENT
In Partial Fulfillment of the Requirements
For the Degree of
MASTER OF SCIENCE
WITH A MAJOR IN ELECTRICAL ENGINEERING
In the Graduate College
THE UNIVERSITY OF ARIZONA

1 9 9 1

STATEMENT BY AUTHOR

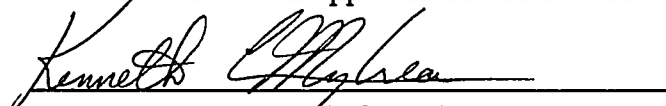
This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the library.

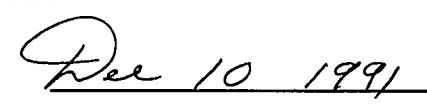
Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgement of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: 

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:


Kenneth C. Mylrea
Professor of
Electrical and Computer Engineering


Date

ACKNOWLEDGEMENTS

I would like to thank my thesis committee for providing the guidance and knowledge necessary for completing my research. Richard Watt, my research advisor, gave me the encouragement and direction I needed, and provided valuable comments during drafting of this thesis. Dr Kenneth Mylrea, my thesis director, originally introduced me to the field of Biomedical Engineering, and for that I am grateful. Dr Bobby Hunt gave me my first structured lessons in Neural Networks and sparked my interest in their application to the medical field. Thank you, again, gentlemen.

Let me also extend thanks to Sandi Sledge, who provided guidance as to my degree requirements and spent many hours checking my thesis format to ensure it was acceptable and professional.

In addition, I would like to acknowledge the encouragement, guidance, and friendship of the following people, who I list only by initials because they would be embarrassed by the recognition: B.B. & J.B., J.K. & S.K., P.W. & M.W., R.S., M.M., G. "Red" R., B.J., and last, but never least, D.M.

This Thesis is dedicated to my family, particularly my father, who always encouraged me to better myself through higher education, and who inspired me with the simple phrase: "Yes, I can."

Namaste!

2.2.2.2	<i>Number of Middle Layer Neurodes</i>	33
2.2.3	<i>BPN Training Parameters</i>	34
2.2.3.1	<i>Learning Constant, η</i>	34
2.2.3.2	<i>Momentum</i>	36
2.2.3.3	<i>Activation Function, $f(\text{net})$</i>	37
2.2.4	<i>BPN Performance</i>	39
2.2.4.1	<i>Training</i>	39
2.2.4.2	<i>Classification</i>	41
2.2.5	<i>A Final Consideration</i>	42
3.0	RESEARCH METHODOLOGY AND IMPLEMENTATION	43
3.1	<i>EEG Data Acquisition and Preprocessing</i>	43
3.1.1	<i>Data Acquisition</i>	43
3.1.2	<i>Front-end Processing</i>	44
3.1.2.1	<i>Discrete Fourier Transform</i>	46
3.1.2.2	<i>DFT Power Spectrum and Phase</i>	46
3.1.2.3	<i>Data Smoothing and Normalization</i>	47
3.1.2.4	<i>BPN Input File Formation</i>	50
3.2	<i>BPN Training Simulations</i>	50
3.2.1	<i>Simulation Programs</i>	51
3.2.2	<i>Network Initialization</i>	51
3.2.3	<i>Topology Variation Methods</i>	53
3.2.4	<i>Parameter Variation Methods</i>	55
3.2.5	<i>Performance Criteria Variation Methods</i>	57
3.2.6	<i>Training Performance Recording Methods</i>	57
3.3	<i>BPN Classification Simulations</i>	58
3.3.1	<i>Simulation Programs</i>	58
3.3.2	<i>Parameter Variation Methods</i>	58
3.3.3	<i>Performance Criteria Variation Methods</i>	59
3.3.4	<i>Classification Performance Recording Methods</i>	59

	6
3.4 Statistical Analysis of EEG Data	59
4.0 RESULTS AND DISCUSSION	61
4.1 BPN Training Simulations	61
4.1.1 <i>Results</i>	61
4.1.1.1 <i>Data Type — Power, Phase, Combined</i>	61
4.1.1.2 <i>Group1 vs. Group2</i>	62
4.1.1.3 <i>Middle Layers</i>	62
4.1.1.4 <i>Middle Layer Size(s)</i>	63
4.1.1.5 <i>Multiple Network BPN</i>	73
4.1.1.6 <i>Learning Constant — η</i>	73
4.1.1.7 <i>Momentum Constant — α</i>	85
4.1.1.8 <i>Activation Function Constant — θ</i>	85
4.1.1.9 <i>Training Performance Criteria</i>	96
4.1.2 <i>Discussion (Implications)</i>	96
4.2 BPN Classification Simulations	102
4.2.1 <i>Results</i>	103
4.2.1.1 <i>Data Type — Power, Phase, Combined</i>	103
4.2.1.2 <i>Group1 vs. Group2</i>	103
4.2.1.3 <i>Middle Layers</i>	104
4.2.1.4 <i>Middle Layer Size</i>	104
4.2.1.5 <i>Multiple Network BPN</i>	104
4.2.1.6 <i>Activation Function Constant — θ</i>	104
4.2.1.7 <i>Training Performance Criteria</i>	105
4.2.1.8 <i>Classification Performance Criteria</i>	105
4.2.1.9 <i>Learning and Momentum Constants</i>	105
4.2.2 <i>Discussion (Implications)</i>	106
4.3 SPSS Analysis	109
4.3.1 <i>Results</i>	109
4.3.2 <i>Comparisons with BPN Results</i>	109
5.0 CONCLUSIONS AND FUTURE STUDY	117

REFERENCES 121

LIST OF FIGURES

Figure	Page
2.1 Sigmoid Function for Three Values of θ	28
2.2 Derivative of the Sigmoid Function for Three Values of θ	38
3.1 Sampled EEG for One Subject at Each MAC Level	45
3.2 EEG Power Spectrum	48
3.3 EEG Phase vs. Frequency	49
3.4 Input Patterns for Each Group	52
3.5 BPN Topologies	54
3.6 Multiple Network BPN	56
4.1 Weight Magnitude vs. Frequency for Neurode 1 of the First Middle Layer for Networks Trained with Phase Data	64
4.2 LMS Error vs. Iterations for Networks Trained with Phase Data	66
4.3 Weight Magnitude vs. Frequency for Neurode 1 of the First Middle Layer for Networks Trained with Power Data	68
4.4 LMS Error vs. Iterations for Networks Trained with Power Data	69
4.5 Iterations vs. Middle Layer Size for Phase-trained BPNs for Different Combinations of η and α	70
4.6 Iterations vs. Middle Layer Size for Combined Data-trained BPNs for Different Combinations of η and α	74
4.7 Weight Magnitude vs. Frequency for Middle Layer Neurode 1 for Networks Trained with Power and Phase Data Alone, and With Combined Data	76
4.8 Average Iterations vs. η for Phase Data for Different Combinations of θ	79
4.9 LMS Error vs. Iterations for Phase Data Training	86
4.10 Weight Magnitude vs. Frequency for Middle Layer Neurode 1 for Phase Data Training	87
4.11 LMS Error vs. Iterations for Power Training	88
4.12 Weight Magnitude vs. Frequency for Middle Layer Neurode 1 for Power Training	89
4.13 Average Iterations vs. α for Phase Data for	

	9
Different Combinations of θ	90
4.14 Iterations vs. Various Combinations of θ	97
4.15 Average Iterations vs. η and α for $\theta_2 = 5.0$ and $\theta_3 = 0.2$ for Power Data	98

LIST OF TABLES

Table	Page
4.1 SPSS Discriminant Analysis (DA) Phase Classification Results, Group1 → Group1	110
4.2 SPSS DA Phase Classification Results, Group1 → Group2	110
4.3 SPSS DA Phase Classification Results, Group2 → Group2	111
4.4 SPSS DA Phase Classification Results, Group2 → Group1	111
4.5 SPSS DA Power Classification Results, Group1 → Group1	112
4.6 SPSS DA Power Classification Results, Group1 → Group2	112
4.7 SPSS DA Power Classification Results, Group2 → Group2	113
4.8 SPSS DA Power Classification Results, Group2 → Group1	113
4.9 SPSS DA Combined Classification Results, Group1 → Group1	114
4.10 SPSS DA Combined Classification Results, Group1 → Group2	114
4.11 SPSS DA Combined Classification Results, Group2 → Group2	115
4.12 SPSS DA Combined Classification Results, Group2 → Group1	115
4.13 Comparison of SPSS Discriminant Analysis with BPN Analysis for EEG Data	116

ABSTRACT

Accurately and consistently determining depth of anesthesia during surgical procedures is a significant problem. A more objective technique than traditional methods is required. Different concentrations of anesthetic drugs have been shown to affect the Electroencephalograph; results, however, are inconsistent when using only visual inspection of the EEG. An automated technique using Neural Networks for classifying anesthetic depth from EEG data is proposed. Neural Networks are reviewed. Reasons for choosing a Backpropagation Network (BPN) are discussed. Ambiguities in previous BPN research are presented. Over 3,000 networks are formed, demonstrating training and classification properties while altering network topologies, parameters and performance criteria. Tests are performed on the Power spectrum and Phase portions of the EEG data. Optimal BPN parameters and topologies are shown. Results are compared with a statistical paradigm.

1.0 INTRODUCTION

The modern operating room has become a conglomerate of technologically sophisticated equipment, staffed by highly trained medical personnel. Yet in spite of all the education and futuristic surgical gear, a patient's health during surgery is still dependent on the relatively subjective "art" of anesthesiology. In an attempt to take the high variability out of anesthesiology, and add a greater degree of objectivity, researchers have proposed several methods for better determining depth of anesthesia. One method is the use of the Electroencephalograph (EEG) to determine brain activity. Although it can be shown that anesthetic drugs do change the EEG, the variability among drug affects is significant. In addition, the amount of EEG data produced can be overwhelming for the anesthesiologist. New computer techniques have eased the burden of manually analyzing this data. A relatively unexplored technique for this application is Neural Networks (NNs). Studies have shown the utility of NNs in pattern recognition tasks such as EEG monitoring. In general, they are equal or superior to their statistical paradigms because of their ability to adapt to highly variable data and handle nonlinearities. The following sections introduce the subjects of anesthesiology, electroencephalography and neural networks.

1.1 Anesthesiology

1.1.1 *Description and History*

Anesthesiology is the practice of administering anesthetic drugs to a patient in order to provide an unconscious and pain free state during surgery [28]. Anesthesia also satisfies the surgeon's need for an immobile body and for relaxed skeletal muscle during certain surgical procedures. The first qualified physician to practice anesthesia was John Snow, who published his introductory manuscript,

On the Inhalation of the Vapour of Ether, in 1847 [35]. Since this time, various inhaled and injected anesthetic drugs have been developed and studied.

All anesthetic drugs require close patient monitoring during surgical procedures in order to maintain an anesthetic concentration that ensures amnesia and analgesia (inability to feel pain). It is common practice to use the lightest general anesthesia required, although this has led to many reports of awareness and occasional pain perception [14, 35]. On the other hand, too high an anesthetic concentration can cause a serious threat to the patient's health and survival [28]. Thomas points out as recently as 1988 the difficulty in reliably predicting patient status [42], and there is no reason to yet believe that circumstances have changed. Because there is no direct means of monitoring anesthetic depth, several indirect methods involving respiration, blood pressure, etc., have been developed.

1.1.2 *Anesthetic Depth Determination*

Since the introduction of anesthesia, classification of anesthetic depth has been an important research area. Plomey followed Snow in 1847 with a subjective categorization method for ether, and later improved on this with a more objective test [40]. Plomey's scheme was briefly replaced in World War I with a general classification for anesthetics in addition to ether. This new scheme served as the basis for Guedel's refined methodology that he first submitted in 1920 and then presented in near complete form in 1937 [35, 40, 42]. Guedel's methods relied heavily on the character of respiration and the presence of muscular relaxation, yet the more recent introduction of neuromuscular blocking agents often makes these determinants useless [2, 42]. Indeed, we now recognize that even studying the potency of anesthetics by using classical methods has several disadvantages [18]. As Navabi pointed out, with surgery becoming increasingly more complex and patients developing more complicated problems, the anesthesiologist can no longer rely on traditional techniques to adequately monitor the patient's status [28].

Researchers have improved on Guedel's classification methods by studying blood pressure, heart rate, muscle relaxation, respiration, sweating and lacrimation (secretion of tears) [40]. Blood pressure does decrease with increasing depth of anesthesia *with some drugs*. Other surgical factors, however, also influence blood pressure, making this method unreliable. Heart rate is a poor indicator of depth because of the variability between drugs. Muscle relaxation, as previously stated, was widely used in determining anesthetic depth since muscle tone was a useful indicator. The introduction of neuromuscular blocking agents to prevent patient movement has rendered this monitoring method mostly ineffective. These blocking agents are also the reason that respiration no longer plays a major role in depth determination. Finally, sweating and lacrimation may occur during light anesthesia, but other surgical stimulants may cause these phenomena, too, even during adequate anesthesia [40].

Newer techniques for determining anesthetic depth are reviewed by Sebel [40]. These include oesophageal contractility and electromyography. Oesophageal contractility may become a useful indicator since the oesophagus is composed of smooth, rather than striated, muscle. Neuromuscular blocking agents only affect striated muscle (the reason why the heart continues beating during administration of these drugs). Upper and lower oesophageal contractility and rate of contractility are being studied. Some possible drawbacks to this approach are the use of smooth muscle relaxant anesthetics and diseases of the oesophagus that affect contractility.

Electromyography (EMG) is useful since light anesthesia is associated with an increase in muscle activity. However, as indicated earlier, neuromuscular blocking agents will partially suppress the EMG. The frontalis muscles in the forehead will generally maintain increased EMG activity under light anesthesia. Sebel warns that this may also be an indicator of return of skeletal muscle activity. As a result, neuromuscular blockade must be monitored concurrently on a separate peripheral nerve in order to achieve accurate interpretation [40].

Discussion of the requirement for adequate patient monitoring and the relative usefulness (uselessness) of traditional techniques point to the need for an objective and reliable technique for judging anesthetic depth. One new approach

which might meet these criteria is Electroencephalography. Because different anesthetic drugs have varied effects on the EEG, an automated process that can account for these variabilities would be useful.

1.1.3 *EEG in Anesthetic Depth Determination*

Electroencephalography is the study of the electrical activity of the brain. The electroencephalograph provides a time domain representation of this electrical activity. The EEG may be recorded noninvasively by attaching electrodes to the scalp, or invasively by using either cortical electrodes on the exposed surface of the brain, or depth electrodes which measure activity within the brain [36]. The invasive (subdermal) technique is more precise because of its proximity to the location of the activity, but requires more than a simple clinical procedure.

The 10–100 μV amplitude electrical signal (10 mV for subdermal EEG) is a complex, nonrepetitive waveform [40]. The spectrum of activity is between 0.5 and 100 Hertz (Hz), but the useful range is usually limited to the delta, theta, alpha, and sigma, and occasionally the beta, frequency bands (0.5–3.5 Hz, 4–7.5 Hz, 8–11.5 Hz, 12–14.5 Hz and 15–30 Hz, respectively) [11, 12, 36]. These bands show different levels of activity depending on the subject's status. Activity, rest, concentration, tension, stress, sleep, and anesthesia all cause a change in specific bands.

Pyramidal cells within the cortex of the brain produce the EEG signal. These cells (neurons) can be interconnected to thousands of other cells by their receiving sections — dendrites — and their transmission sections — axons. During normal activity, the pyramidal cell maintains a specific electro-chemical potential (polarization) across its membrane. High levels of activity in the dendrites of a cell can cause the local depolarization of the dendrite. This depolarization creates a travelling charge along the dendrite's membrane. If enough dendrites become depolarized, an *action potential* is created in the axon. The axon transmits this action potential to other pyramidal cells' dendrites, where the process can happen again. It is these action potentials that are recorded by the EEG [36].

Berezowskyj *et al* trace the first study of anesthetic effects on EEG to a report by Gibbs *et al* in 1937, and the first practical determination of anesthetic depth by EEG to Courtin *et al* in 1950 [2]. It was noted that, generally, under increasing levels of anesthesia, the EEG initially speeds up in activity and then becomes slower in frequency and higher in amplitude (similar to sleep patterns), although different anesthetics produced different EEGs [18, 35, 40, 42]. Scott *et al* note that the EEG is particularly useful because it changes in a predictable manner under given agents [39].

For many years after Courtin's observations, gross visual interpretation of the EEG was attempted with some success. This type of analysis, though, required a great deal of skill in interpreting the time-domain recordings, and was subject to a high degree of variability between electroencephalographers [2, 24]. Additionally, because of the length of a typical surgical procedure, it was, and still is, difficult for the anesthesiologist to maintain vigilance on the raw EEG information. Sebel insists that an adequate anesthetic depth monitor based on conventional paper EEG recordings has yet to be found [40], and Levy *et al* state that, although EEG information may be sufficient to categorize even small depth changes, these changes are too small to be noticed through routine visual inspection [24]. (Besides the potential application to determining depth of anesthesia, the EEG is used for other purposes during surgery. Electroencephalography is a recognized indicator of hypoxia. Therefore, EEG monitoring has been recommended for a variety of surgeries, including cerebrovascular, cardiopulmonary bypass and deliberate hypotension [24].)

In the 1970's and '80's, electroencephalographers began concentrating on the power spectrum image of the EEG in an attempt to better understand and use their recordings. Power spectrum analysis is useful for several reasons: it retains much of the information present in the original signal (although phase is lost); it allows classification of several distinct brain wave types; and, it simplifies the identification of small changes in the EEG. Additionally, EEG recorders introduce low-frequency baseline drift, which may be eliminated by combining these components into one frequency bin [24]. Without the benefit of automated processing, power spectrum

analysis is difficult for two reasons: (1) It is still necessary to maintain vigilance of patient status over the entire length of the operating procedure, and (2) small changes in low amplitude signals that may signify important status changes can be swamped by high amplitude signals. Several researchers have used power spectrum analysis of the EEG, coupled with powerful automated processing techniques, to objectively and efficiently determine anesthetic depth *under certain conditions* [2, 14, 23, 24, 36, 39, 40]. Levy emphasizes the specificity of these previous tests [23]. A truly useful technique for EEG monitoring would be applicable to a wide variety of conditions. It is possible that the adaptability of Neural Networks could fill this requirement.

1.2 Introduction to Neural Networks

1.2.1 What is a Neural Network?

The term “Neural Network” conjures up many different definitions in people’s minds. This is understandable, since these days it could refer to either the natural, biological nervous system of all animals, or to an emerging field in computer technology. For our purposes, Neural Networks (NNs) refers to the latter. Sometimes it is preferred to call computer NNs “Artificial Neural Networks”, or any of a host of specially coined terms. As long as the reader knows what field is being discussed, NNs is sufficient. Later, the term *Neural Network Simulations* will be introduced to refer to computer programs that simulate NNs, since a physical NN is impractical as of today [5].

In its simplest definition, a Neural Network is an approximation of the processing characteristics of the mammalian nervous system. It is an attempt to use a parallel, highly interconnected nervous system model, and have that model *learn* how to perform a task. Our current Von Neumann computer architectures, performing in their traditionally serial mode, are quite good at solving well defined, linear problems (“linear” referring to following one, straight path of a problem flow chart from beginning to end). NNs have the ability to solve abstract problems, where an actual algorithm for a solution is unknown or poorly understood.

NNs are composed of two primary elements: neurodes and interconnections. The neurode approximates a biological neuron and the interconnections approximate the dendrites, axons and synapses of the neuron. How neurodes are specified, how they are connected, and how the interconnections are altered during “learning” characterize the type of NN being used.

Work in this area goes back over 40 years when first attempts were made to construct mathematical models of nervous system behaviour [25]. Lack of practical training algorithms created a lull in the field for several years. New research in the early 1980’s rekindled interest and now a Neural Network boom, as evidenced by over a dozen international conferences in this area in the past five years alone.

Caudill [5] and Lippman [25] give excellent overviews of Neural Network characteristics and history, and Bailey and Thompson [1] review important distinctions between NN types. Section 2.0 discusses in more detail the characteristics of NNs.

1.2.2 *The Neural Network Role in Pattern Recognition*

1.2.2.1 *Comparison with Statistical Pattern Recognition*

As stated previously, NNs are best suited to abstract problems where a solution algorithm is unknown or poorly understood. This is a classical characteristic of pattern recognition. How does one extract relevant features from a pattern, or, more importantly, what are the relevant features? What decision rule will be used to judge class membership? How are new patterns classified when they are missing information or contain more information? Statistical pattern recognition techniques, such as the Bayesian approach, rely heavily on being able to construct a decision rule based on large amounts of unambiguous data, and attempt to minimize overall cost of errors [32]. Once the decision rule is formed, it is static. Contrast this with the NN, which has the ability to adapt to new data, compensates for minor variabilities, and can process large or small amounts of data using very simple processing elements.

Certain NNs are drawn from their statistical paradigms. A formal equivalence between multilayer perceptrons (MLP) (see section 2.2.1) and discriminant analysis (DA) has been shown by Gallinari *et al* [9], while Horne & Hush assert the equivalence of this method to Bayesian classification [17]. DA utilizes Bayes' *a posteriori* probability in order to classify inputs. Regardless of the contributions of NNs, the importance of statistical pattern recognition for linearly separable data should not be overlooked. In fact, Rose demonstrates good results when implementing discriminant analysis on EEG data [36].

1.2.2.2 *Neural Network Pattern Recognition*

NNs show their real power when decision surfaces become nonlinear [9]. It seems appropriate that a model of mammalian intelligence would be better suited to distinguishing patterns: It is easy for a human to distinguish between species of birds, or even their sexes, and yet difficult for a computer. Thus, pattern recognition has become one of the most researched NN applications. The enormous amounts of processing power required in pattern recognition suggests a need for a new technique, and NNs answer that need. Pao [32] is recommended for information on the motivation behind pattern recognition and a history of statistical and computer-based pattern recognition.

1.2.3 *Previous Clinical Applications of Neural Networks*

The medical field is a logical location for NN research. The tremendous amounts of physiological data obtained from patients, coupled with the need for objective, efficient processing and classification fit neatly with NN capabilities. Egbert *et al* have successfully used NNs in thermal infrared image recognition to distinguish tumors in laboratory mice [8]. Meyer *et al* have identified hydrogen nuclear magnetic resonance spectra of complex oligosaccharides [27]. Navabi uses NNs as the basis for a smart alarm system in the operating room [28]. Rayburn *et al* demonstrate an NNs ability to identify cardiovascular waveforms [33]. Schizas *et al* show excellent results using NNs to classify electromyographic signals [38]; Gevins *et al* are pioneers in the use of NNs in EEG evaluation [10, 11, 12]. These

are but a few of the clinical applications of NNs *in pattern recognition alone*. Other fields, including control of prosthetics, are also seeing their share of NN research.

1.2.4 NN Parameter Selection Problems

If NNs are the best way to perform pattern recognition, then why aren't Neural Networks more widely used? One answer is, of course, the newness of the field. Another significant reason is the problem with selecting the appropriate NN and then selecting the network parameters that will ensure efficient, accurate solution of the problem at hand. This is no easy task. Much of the research in this field falls in this area (witness the thirty years from the introduction of the first NN math models to the practical application of an NN in the 1980's) [1, 5, 20, 25, 26, 32]. While there is a tremendous amount of research in this area, much of it is contradictory or ambiguous. (Higashino *et al* [16] and Gorman *et al* [13] use learning constants that are proven to be unstable; Izui and Pentland [19] state that convergence speed depends directly on an activation function constant that Caudill [5] says is unimportant; Pao [32] contradicts himself and Rumelhart *et al* [37] when adding momentum to his weight update equations, and later presents a NN simulation program which makes a second contradiction. See section 2.2 for more information on the parameters mentioned above and additional discussion on the contradictions.)

The typical person desiring to implement a simple NN is quickly bogged down in choosing between *supervised vs unsupervised learning*, *number of neurodes*, and *activation functions*. It is also true that much of the research performed to eliminate these problems is extremely data dependent, often relying only on training data constructed manually rather than data collected empirically [7]. Indeed it would only be prudent that any future research contain a caveat stating whether findings are true in general or specific to the input data utilized. I would have to agree with Schizas *et al* [38] that there is very little information available that helps in choosing important network parameters. Because of contradictions and ambiguities, it is difficult to recommend any one source for help. A more detailed

study of what happens when various network topologies and parameters are used is needed.

1.3 Research Goals

Based on the information presented above, three goals are attempted for this research:

- Present a detailed account of NN training and testing characteristics for EEG data based on various topologies and network parameters. This will attempt to confirm previous research referenced in this thesis, develop new areas of consideration, and answer some of the ambiguities and vagueness in the literature.
- Propose an optimal, feasible NN for electroencephalography classification. This NN should be expandable, both to larger data sets and to different types of data.
- Compare and contrast the optimal NN with its statistical paradigm. Is the NN indeed superior to its statistical counterpart for this problem?

2.0 NEURAL NETWORKS — BACKPROPAGATION NETWORKS

Neural Networks are powerful tools for solving problems with poorly defined solution algorithms. Many different types of NNs exist, all characterized by their topologies and learning rules. Multi-layer feed-forward networks using backward error propagation (commonly referred to as Backpropagation Networks (BPNs)) have emerged as one of the most widely used NNs because they have a wide range of applications and they are relatively simple to implement. There are, however, several features of the BPN which are difficult to characterize. These features are important to the training and classification capabilities of the BPN. The next sections give more detail about various Neural Networks, discuss why BPNs are optimal for EEG processing, and present a study of BPN topology, algorithms, and important parameters.

2.1 Neural Networks

2.1.1 *Summary of NN Characteristics*

NNs are characterized by their topologies and learning rules [1, 5, 25]. As discussed earlier, the topologies can be composed of a single neurode or thousands of neurodes, interconnected in a variety of manners. The neurodes themselves are simple computational elements. Learning rules specify how the neurode interconnections should be changed during training in order to achieve a desired solution. These characteristics of NNs attempt to model the highly parallel, highly interconnected mammalian nervous system in order to solve complex tasks such as pattern recognition.

In pattern classification tasks, the general NN can be broken into two stages:
(1) A *pattern matching stage*, where the current input pattern is compared with

previous patterns, and an intermediate score is output, and (2) a *selection and enhancement stage*, where the intermediate scores are classified and the results are used to modify how the pattern matching stage computes its intermediate scores [25]. The output of the second stage may be the final output of the NN, or it may connect to additional stages. In the most general case, a single neurode with several inputs will contain both stages and have one output line [5]. This neurode will train its input lines to recognize and enhance important input features, and then indicate the pattern class by setting the appropriate output (i.e. “0” for class 1 and “1” for class 2). (Pao [32] gives a more general definition of natural vs. computer-based pattern recognition techniques, including feature extraction and classification methods. Kohonen [21] presents an introduction to applications of neural computers to pattern recognition.)

Lippman presents a taxonomy of six NNs that can be used as classifiers [25, Figure 3]. This taxonomy first divides the nets by input type: binary vs. continuous. Binary inputs are useful where patterns may be represented easily using a “0” or “1” for features (such as a 5x7 bit map of alphanumeric characters with no grey-scale). This is opposed to continuous inputs, which are generally more useful since they allow more variability in the data. Additionally, most of the NNs that specify continuous inputs will accept binary input, while the converse is not true.

The next level of Lippman’s taxonomy separates networks by training method: supervised vs. unsupervised. In supervised learning, class membership is already known and is used to reinforce the desired output. In unsupervised learning, the NN is allowed to create its own classes, resulting in clustering of the input patterns. Bailey *et al* state that, in supervised learning, the desired output may be difficult to obtain and training time is of moderate length, while unsupervised learning time is generally comparatively short [1].

The final level of the taxonomy specifies a NN that meets the previous requirements. Lippman also shows the classical paradigm for each of the general NNs (see section 1.2.2.1). (Lippman [25] and Caudill [5] give excellent background information on various NNs.)

2.1.2 *Choosing an Optimum Classifier for EEG Data*

While an algorithm for solving a particular pattern recognition technique may be unknown, certain characteristics about the data type and availability, the expected output, and the eventual implementation environment of the NN are usually known. These features are true of EEG data in the operating room. It is these known characteristics that will eventually specify the type of NN to use. For example, EEG data is continuous: It has already been shown that continuous data input specifies a particular subset of NNs from those that accept binary data (section 2.1.1).

Bailey and Thompson's criteria [1, Tables 1 & 2] serve as an excellent means for selecting an optimal NN. With these criteria in mind and examining the task of using EEG signals in the operating room to determine depth of anesthesia, the following data characteristics and required NN capabilities can be stated:

- The input EEG data is continuous, complex, and has a high information content.
- Output is a specified level of anesthesia, and can be represented using binary groups.
- Network training can be performed off-line; therefore, training time can be slow.
- Classification should be fast in order to present real-time analysis of patient status.
- The final network will have a high utilization.

Only multi-layer feed-forward networks with backward error propagation (commonly referred to as Backpropagation Networks (BPN)) and Counterpropagation Networks (CPN) in Bailey and Thompson's tables fit all the criteria. The Madaline paradigm fits most of the required capabilities, but it only takes bipolar input. Thus, a BPN or CPN appears to be appropriate for this problem. Because of the extensive amount of literature on the BPN (owing to its longer history), this was the network chosen. It was felt that the number of previous studies in

BPN performance would be useful in comparing and contrasting results from this research.

2.2 Backpropagation Networks

2.2.1 *Theory and Developmental History*

Backpropagation Networks draw on two main bodies of research: the perceptron and the Widrow-Hoff or LMS algorithm. Perceptrons were first introduced by Frank Rosenblatt in 1959 and then further reported on by M. Minsky and S. Papert in 1969 [5, 25, 32], although Minsky and Papert concentrated on the limitations of the perceptron scheme. The perceptron is one of the simplest neurodes developed. Inputs to the perceptron arrive on weighted input lines. The perceptron sums these weighted inputs and passes the result to a non-linear activation function (AF). The AF determines if the perceptron output is high (1) or low (-1 or 0). Rosenblatt developed the first convergence algorithm for the perceptron that adjusted the weights dependent on the output error. This simple NN provoked some interest, but few applications resulted since it could only solve problems that were linearly separable [5, 25, 32].

If single layer perceptrons could form decision regions for linearly separable problems, then it seemed likely that multi-layer perceptrons (MLPs) could solve the more difficult non-linearly separable kind. Indeed, MLPs overcame the problems of single layer perceptrons, but, due to a lack of adequate training algorithms (Rosenblatt's Perceptron Convergence procedure just would not work), they were of little use [25]. Then the Widrow-Hoff (LMS) algorithm was introduced. This procedure minimizes the mean squared error between the target output and the actual output. It also requires that the non-linear activation function be replaced with a continuously differentiable AF, such as a sigmoid [25] (Widrow and Stearns present various methods for adaptive signal processing, including the LMS algorithm, in [44]). The sigmoid forms smooth boundaries between regions, and is differentiable over its entire range [1], an important characteristic that will be discussed in detail later. (Jordon discusses the use of a sigmoid as a neurode AF [20].

He coins the term *semi-linear* to refer to a linear system modified to produce a non-linear system, but which still has a linear component. This term has been used to refer to Backpropagation Networks as “semi-linear feed-forward nets”.) Rumelhart *et al* combined all this research to develop the Generalized Delta Rule (GDR) for updating neurode weights, and this in turn became the basis for Backpropagation Networks [37] (Kohonen actually attributes the invention of backpropagation of errors to work by Werbos in 1975 and '82 [21]). It has been shown that a BPN with three *functional* layers is capable of forming any arbitrarily complex decision region [5, 25, 37].

Backpropagation Networks are hierarchical networks: that is, they have at least three layers: An input (fan-out) layer, a middle (or hidden) layer, and an output layer (the discussion in section 2.2.2.1 focuses on the need for more than one middle layer). Generally, all the neurodes in a given layer are connected to all the neurodes in the next layer. The network is trained by first passing inputs forward through the layers and forming outputs. The input to a given neurode is the weighted sum of the outputs from the previous layer, except for the first (input) layer, which receives its inputs directly [32]. At the output layer, the LMS error between the target and actual output is calculated and then “Backpropagated” through the network in order to adjust the connection weights. This continues until the error is minimized (usually some ϵ where $\epsilon > 0$, since it would take an infinite amount of time for the error to reach 0).

BPN weights are adjusted according to the LMS/Generalized Delta Rule. This rule finds the minimum LMS output error by taking the partial derivative of the error with respect to the neurode weights. Thus, this algorithm is a *gradient descent* method for finding the network solution. As such, the weights and error specify a hyperparaboloid in n -space (actually just some upside-down curved surface with hills and valleys). The negative gradient of the surface specifies the vector pointing in the direction of the most efficient path (at a given point) to the surface’s minimum point [5, 22]. Moving down this surface by adjusting the weight values leads to the minimum.

The weighted input to neurode j in a layer is

$$\text{net}_j = \sum_{i=1}^m w_{ji} x_i \quad (2.1)$$

where w_{ji} is the weight of the i th input line to neurode j , x_i is the value of the i th input, and m is the number of inputs to neurode j . The initial values of the weights are usually set to random values in the range $[-1 \dots 1]$. If the weights are all initially the same, the network will be unable to train [5, 25, 32, 37], which will be evident later. The output of neurode j is

$$o_j = f(\text{net}_j) \quad (2.2)$$

where f is the activation function.

The LMS algorithm requires a continuous, S-shaped, monotonic AF that asymptotically approaches fixed values [1, 5, 26]. If this AF is a sigmoid, equation (2.2) becomes

$$f(\text{net}_j) = \frac{1}{1 + e^{-\theta \text{net}_j}} \quad (2.3)$$

where e is the exponential constant. In equation (2.3), θ describes the shape of the sigmoid. A high value of θ steepens the slope, while a low value flattens the slope (see Figure 2.1). The sigmoid requires that the variable, net_j , be in the range $[-\infty \dots \infty]$. If the neurode inputs are constrained to be in the range $[0 \dots 1]$ (as they are for layers beyond the first middle layer when the AF asymptotically approaches 0 and 1, and usually are for initial inputs to the network; in Caudill's discussion, the input vector components are constrained to be in the range $[0 \dots 1]$ [5]), then the weights must be able to take on values of $[-\infty \dots \infty]$.

Each input pattern is presented to the network during training and the outputs at the final layer are collected. Once all the patterns have been presented, the mean-squared error is calculated as

$$E = \frac{1}{2P} \sum_{p=1}^P \sum_{j=1}^m (t_{pj} - o_{pj})^2 \quad (2.4)$$

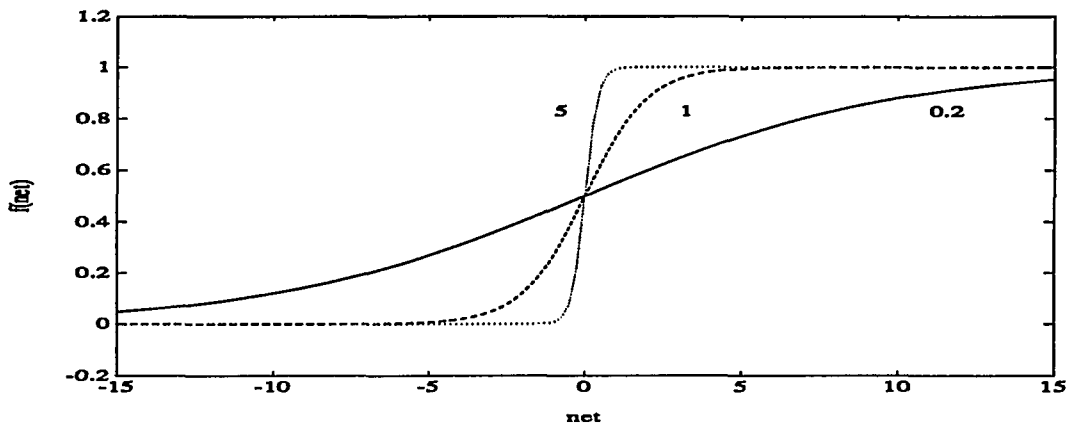


Figure 2.1: Sigmoid Function for Three Values of θ .

where t_{pj} is the target output for neurode j of pattern p , o_{pj} is the actual output of neurode j for pattern p , m is the size of the output layer, and P is the number of input patterns. The factor of one-half is added for mathematical convenience [32].

By computing the negative gradient of (2.4) with respect to the weights, we compute the optimal vector *at a given place on the hyperparaboloid* pointing towards the minimum E [5, 22]. An incremental weight change proportional to this negative gradient is added to the current values of the weights:

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} \quad (2.5)$$

where η is called the learning constant or step size (sometimes referred to as β [5]). From (2.4), we know that E is a function of the output, o , and o , in turn, is a function of a neurode's AF, weights, and input (see equations (2.1) and (2.3)). The input is just the output from the previous layer's neurode(s). Thus, it can be shown that (2.5) can be expressed as

$$\Delta_p w_{ji} = \eta \delta_{pj} o_{pi} \quad (2.6)$$

where the subscript p denotes the pattern number, and

$$\delta_{pj} = (t_{pj} - o_{pj}) f'(\text{net}_{pj}) \quad (2.7)$$

if the neurode is in an output layer, and

$$\delta_{pj} = f'(\text{net}_{pj}) \sum_{k=1}^m \delta_{pk} w_{kj} \quad (2.8)$$

if the neurode is in a middle layer (for a full derivation of these formulas, see Rumelhart *et al* [37] or Pao [32]). The δ 's on the right side of (2.8) are from the next higher layer.

The real beauty of a sigmoid activation function can now be demonstrated. For the AF of (2.3),

$$f'(\text{net}_{pj}) = \theta \cdot f(\text{net}_{pj}) \cdot (1 - f(\text{net}_{pj})) \quad (2.9)$$

or, since $f(\text{net})$ is the output, o :

$$\frac{\partial o_{pj}}{\partial \text{net}_{pj}} = \theta \cdot o_{pj} \cdot (1 - o_{pj}) \quad (2.10)$$

This is useful in implementing the algorithm since all that is needed to compute $f'(\text{net})$ is the previously computed value of $f(\text{net})$. Equations (2.7) and (2.8) may now be expressed as

$$\delta_{pj} = (t_{pj} - o_{pj}) \cdot \theta \cdot o_{pj} \cdot (1 - o_{pj}) \quad (2.11)$$

$$\delta_{pj} = \theta \cdot o_{pj} \cdot (1 - o_{pj}) \sum_{k=1}^m \delta_{pk} w_{kj} \quad (2.12)$$

for the output layer and middle layers, respectively.

In order to be a true gradient descent solution, all the $\Delta_p w_{ji}$'s in equation 2.6 must be calculated first and then added to the current weight values. That is,

$$\Delta w_{ji} = \sum_{p=1}^P \Delta_p w_{ji} \quad (2.13)$$

is the value added to the current weight of the i th input line of the j th neurode [5, 32, 37].

The irregular nature of the Error vs. weights surface can contain many local minima. It is a common problem in gradient descent (and, thus, Backpropagation) that during training, the weights may become “stuck” in a local minimum [5, 32, 37]. In order to overcome this problem, *momentum* is usually added to the equation. Momentum will tend to keep the solution moving in the same direction it took during the last iteration of the algorithm, hopefully moving past local minima. This new term is added by modifying (2.6) and (2.13) to produce

$$\Delta w_{ji}(n+1) = \left(\sum_{p=1}^P \eta \delta_{pj} x_{pi} \right) + \alpha \Delta w_{ji}(n) \quad (2.14)$$

where α is a proportionality constant, and n is the n th step of the algorithm. Pao is ambiguous in the way he states this [32, equations 5.42 and 5.43], while Rumelhart *et al* provide little additional insight [37]. Caudill’s suggestion that the momentum be the average of the weight change over all the patterns from the previous iteration is appropriate [5].

2.2.2 Topological Considerations

Choosing a BPN requires the implementer to consider two important network topological features: The number of middle layers to use, and the number of neurodes in each middle layer [1]. Other topological considerations are fairly straightforward, such as choosing the size of the input and output layers and how to interconnect the neurodes. The size of the input layer is specified by the size of the input vector: one neurode is needed for each vector component. In reality, choosing the input vector size depends on knowledge about the information content, or features, of the input pattern. Fortunately, the NN is superior to its statistical counterparts because of its ability to compensate for poor feature selection [27].

For BPN classifiers, the output layer size is usually coincident with the number of classes. There seems little argument that a separate neurode for each class provides the best training and classification capabilities [5, 25].

In the case of interconnections, it is most appropriate to connect all neurodes from one layer to all neurodes in the next highest layer [5]. Training will eliminate those interconnections that are insignificant by letting the weight approach 0.

2.2.2.1 *Number of Middle Layers*

Studies attempting to determine the optimum number of network middle layers trace their roots back to the very beginnings of NN research. As reported in section 2.2.1, a three-layer network (two middle layers and an output layer, where the input layer is simply a fan-out) can solve a given non-linearly separable problem. According to this theory, the network can be examined in the following manner. The first-middle layer forms the hyperplanes (hypersurfaces in n -space, although in any space the boundary is not really straight but curved for a sigmoid AF; see [25, Figure 16] for a demonstration) specified by the input data. The second-middle layer performs a logical AND of the hyperplanes in order to form decision regions. Finally, the output layer performs a logical OR on the regions if separate regions specify the same class [25, 32] (this would be unnecessary if the problem were linearly separable).

Still, the necessity of a second middle layer became suspect. In 1987, Robert Hecht-Nielsen recognized the applicability of a theorem by a mathematician named Andrei Kolmogorov to the NN field [15]. This theorem guaranteed that a layered network (one with only three *total* layers of neurodes) existed that could solve a given non-linearly separable problem. (As a sidenote, Lorentz in 1976 observed of Kolmogorov's theorem that it was "of the nature of a pathological example, whose main purpose is to disprove hopes that are too optimistic" [26]. Eleven-years later, Hecht-Nielsen discovered the theorem's significance to neurocomputing.) But Hecht-Nielsen noted that parameters necessitated by Kolmogorov's theorem made its usefulness doubtful, at least for the time-being. This is discussed further in the next section.

Caudill introduces some ambiguity to the problem of selecting number of middle layers [5]. She applies Hecht-Nielsen's observations to the classic XOR

function, identifying this as a non-linearly separable function. This is a slight misapplication of the term “non-linearly separable”. The XOR function *is* linearly separable, it just requires *two* hyperplanes to perform the separations. A truly non-linearly separable problem is one where there are overlapping convex regions. Caudill’s assumptions are similar to Ken-ichi Funahashi’s, who proves that a network with one middle layer can approximate any continuous mapping. Chester [7] points out that Funahashi makes several assumptions about the problem at hand that are violated by most practical problems. In the end, a net with only one middle layer would have to be allowed to grow without bound to solve any mapping problem.

Pao provides another perspective by presenting Nilsson’s 1965 viewpoint on network formation [32]. Instead of the AND-OR functions of the network as stated above, this viewpoint considers the network layers as performing coordinate transformations of the input data. That is, the input pattern space is mapped to another space where the problem is now linearly separable. Like the Hecht-Nielsen approach, this perspective no longer requires two middle layers: in contrast, the solution may be found with less than, or more than, two middle layers. In fact, Gorman and Sejnowski demonstrate some success with a BPN with *no middle layers* [13].

As a practical solution to the problem of selecting the number of middle layers, Bailey and Thompson suggest starting with one middle layer and adding more if needed [1]. While the number of middle layers augments the feature extraction and classification capabilities of a network, it also greatly increases the training time. Caudill agrees with this approach and recommends two middle layers only when there is an overwhelming reason to have more than one [6]. Chester asserts that two middle layers are better than one because the effects of neurons can be isolated, improving weight (and, thus, classification) approximation [7]. None of the relevant literature presents justification for nets larger than two middle layers.

2.2.2.2 *Number of Middle Layer Neurodes*

The number of neurodes in a middle layer greatly determines the training rate and classification capability of the BPN. The general view of the number of neurodes to use is that more neurodes tends to let the network memorize input patterns but allows varied feature extraction, while fewer neurodes tends to make the network generalize, allowing more variation in untrained data inputs [1, 5, 6, 25]. Decreasing the number of neurodes will extend the number of training iterations required [5] and may even prevent the net from converging on a solution [1]. Lippmann's qualitative analysis is that there should be enough neurodes to form a decision region with the complexity to solve the problem, while still small enough that the weight values can be accurately determined from the training data [25].

A more mathematical treatment of the number of neurodes required can be performed using the first network formation perspective given in section 2.2.2.1. Let it be given that the input patterns are of dimension N . The first middle layer will form a hyperplane in N -dimensional space. The second middle layer (AND layer) forms the hyperregions. This layer requires two hyperplanes for each hyperregion, one for each side. Since the hyperplane is N -dimensional, the number of neurodes in the first middle layer is $2N$ [25, 32]. If two or more disconnected hyperregions specify the same class, there must be one-second middle neurode for each of these regions [25]; In the case where hyperregions for different classes are meshed, at least one-second middle layer neurode is needed. Of course, whether hyperregions are disconnected or meshed is usually unknown at the beginning of a problem. Lippmann adds confusion to his analysis by stating that typically there should be "three times as many nodes in the second (middle) layer as in the first layer" [25]. This is unjustified by his previous discussion.

Again pursuing Hecht-Nielsen's observations, Kolmogorov's theorem specifies $2N+1$ neurodes in the middle layer for networks with only one middle layer [15]. Lippmann contradicts this by insisting Kolmogorov's theorem requires $N(2N+1)$

neurodes [25]. Reviewing Lippmann’s source [26] shows that his assertion is incorrect. Caudill states that “in general, the size of the middle layer isn’t a critical parameter” [6], although in an earlier report she does agree with Hecht-Nielsen’s findings [5].

Bailey and Thompson, without elaboration, state that a middle layer that is 75% of the size of the input layer is a suitable starting point when there is only one middle layer [1]. Adding layers while reducing the size of each layer can give similar results, they say.

It should be noted that while adding neurodes may decrease the number of training iterations, it will increase the computational time per each pass. Therefore, the cost of iterations versus time should be examined before adding neurodes to any layer.

Little of the research referenced in section 2.2.2 reported on the classification capabilities of different topological networks trained on empirical data.

2.2.3 *BPN Training Parameters*

BPN training utilizes the equations developed in section 2.2.1. In that section, several terms were introduced: η , momentum and the activation function. These terms require further examination because of their importance to BPN training and because of ambiguities in the relevant literature.

2.2.3.1 *Learning Constant, η*

The learning constant, η , controls the size of the step taken down the gradient towards minimum mean-squared error, as shown in (2.5) and (2.6). Obviously, this value needs to be greater than 0 or the solution will move away from the minimum. Mathematical analysis of gradient descent also shows that η must be in the range $[0 \dots 1]$ or instability will result [5, 6, 25, 44]. Widrow [44] demonstrates this as follows. The input pattern vectors are represented in matrix form as

$$\mathbf{X} = [x_1 x_2 \dots x_P]$$

where x_n is an input pattern column vector, and P is the number of patterns. The covariance matrix is denoted as

$$\mathbf{R} = [\mathbf{X}\mathbf{X}^T]$$

where \mathbf{X}^T is the transpose of \mathbf{X} . Widrow shows that, in order for the gradient descent to be stable and convergent, η must be in the range

$$0 < \eta < \frac{1}{\lambda_{max}} \quad (2.15)$$

where

$$\begin{aligned} \lambda_{max} &\leq \sum(\text{diag. elem. of } \mathbf{R}) \\ &\leq \text{tr}(\mathbf{R}) \end{aligned}$$

where $\text{tr}(\mathbf{R})$ is called the “trace of \mathbf{R} ”. It can be shown that for an input matrix, \mathbf{X} , with values normalized between $[0 \dots 1]$, the trace of \mathbf{R} may be evaluated as

$$\text{tr}(\mathbf{R}) \geq 1$$

and, therefore, (2.15) may be reduced to

$$0 < \eta < 1. \quad (2.16)$$

Flying in the face of this “law” are three researchers: Higashino *et al* [16] use η values as high as 3, 10 and 100 (stating that η be as high as possible “within the range in which the gradient approximation is correct”); Gorman *et al* [13] use $\eta = 2$; and Cater [4] uses initial values of $\eta \geq 10$, reducing this value to < 1 as the error decreases. To accomplish this, he uses an algorithm that adds complexity to the GDR and stores different η 's for each pattern.

Even though these researchers have found success with $\eta > 1$, it is best that the learning constant remain in the range $[0 \dots 1]$. This still leaves an infinite number of choices for the value. A large value of η (near 1) corresponds to a faster learning rate but may lead to oscillations around the ideal solution [5, 25, 32, 37]. Widrow says that it is typical to use an η that is one-tenth of the λ_{max} given in (2.15) [44].

Caudill gives a good, general criteria for choosing η [5]. In this reference, she submits that knowledge of data characteristics can help find an optimum value for η . If the data samples have low variability (i.e. tight clustering), then a high η may be used. Scattered data samples require a lower η in order to find relevant distinctions in the data. In practice, these data distinctions may be difficult to find, particularly when the pattern dimension space is large. Bershad [3] says that the optimal η decreases as the number of training patterns increases, and that it is very dependent on the network's initial weights (among other properties peculiar to his analysis).

2.2.3.2 Momentum

As stated in section 2.2.1 and shown in (2.14), momentum is useful in keeping gradient descent problems out of local minimum. Several researchers emphasize the ability of momentum to decrease oscillations while simultaneously increasing training rate [5, 6, 25, 37], although Pao says that it serves to slow the rate. One lone dissent is voiced by Von Lehmen *et al* [43], who feel that the extra memory required by adding a momentum term outweighs its usefulness to training.

Although *most* researchers agree on the advantages of momentum, there is little agreement, or discussion, on the value of the momentum constant, α . A mathematical necessity is that α lie in the range $(0 \dots 1)$ [25]. This can be proven as follows. Rewrite equation (2.14) as

$$\Delta w(n+1) = d(n) + \alpha \Delta w(n) \quad (2.17)$$

where $d(n)$ is the driving force (neurode subscripts have been dropped for convenience). Rearranging (2.17),

$$\Delta w(n+1) - \alpha \Delta w(n) = d(n) \quad (2.18)$$

The Discrete Fourier Transform of (2.18) is

$$uW(u) - \alpha W(u) = D(u)$$

or, rearranging,

$$W(u)[u - \alpha] = D(u)$$

The transfer function is just

$$\frac{W(u)}{D(u)} = \frac{1}{u - \alpha} = \frac{u^{-1}}{1 - \alpha u^{-1}} \quad , \quad u = e^{j\theta} \quad (2.19)$$

which has a zero when $u \rightarrow \infty$, and a pole at $u = \alpha$ (here, θ is the phase angle, not the activation function constant). Because the coefficient of the denominator of (2.19) is “1”, the pole-zero diagram may be plotted as the unit circle. For stability, α must lie within the circle. Therefore $\alpha < 1$. Since the coefficients of the original weight equation are all real, there are no complex conjugates. Thus it can be concluded that α lies on the real axis. In addition, since we desire to remove high frequency oscillations in the weight update equation, we are, in effect, designing momentum to be a low-pass filter. To accomplish this, the vector expressed by $1 - \alpha u$ on the pole-zero plot must increase as θ increases. This can only happen if αu lies on the positive half of the real axis. Therefore, $0 < \alpha < 1$!

The only other relevant reference to momentum found in the literature was by Higashino *et al* [16] who state without proof that the number of iterations to convergence was proportional to $1 - \alpha$.

2.2.3.3 Activation Function, $f(\text{net})$

Every neurode determines its activation state by using an activation function. For the purposes of the Generalized Delta Rule, this function must be continuous, monotonic, asymptotically approach high and low values, and be continuously differentiable over its entire range [5, 20, 25, 26, 37]. The sigmoid function introduced for this application by Jordan [20] meets these requirements. Several studies have used different functions, but the sigmoid enjoys the most research. To date, it is the simplest AF to implement, as demonstrated in equations (2.9) and (2.10).

The sigmoid in equation (2.3) has several interesting properties. First, as the neurode input, net , approaches $-\infty$, the output approaches 0, as shown in Figure 2.1. The neurode is effectively “off”, contributing nothing to subsequent layers.

Second, as the input approaches ∞ , the output approaches 1. This indicates the highest level of activity for the neurode. Third, when the input is 0, the output is .5, indicating the highest degree of uncertainty in the neurode.

The derivative of the sigmoid, f' , equations (2.9) and (2.10), shows additional interesting properties. First, if the neurode input is approaching either $-\infty$ or ∞ , producing an output of 0 or 1, respectively, then f' approaches 0, as shown in Figure 2.2. This indicates that the weight change for this input will be very small (i.e. the neurode has probably made a good estimate as to the value of the weights and should not change them). Second, if the neurode input is 0, producing an output of .5, then f' will be $.25\theta$, indicating the highest degree of training required (i.e. the neurode has not made a good estimate as to the correct value of the weights and should make the maximum allowable change).

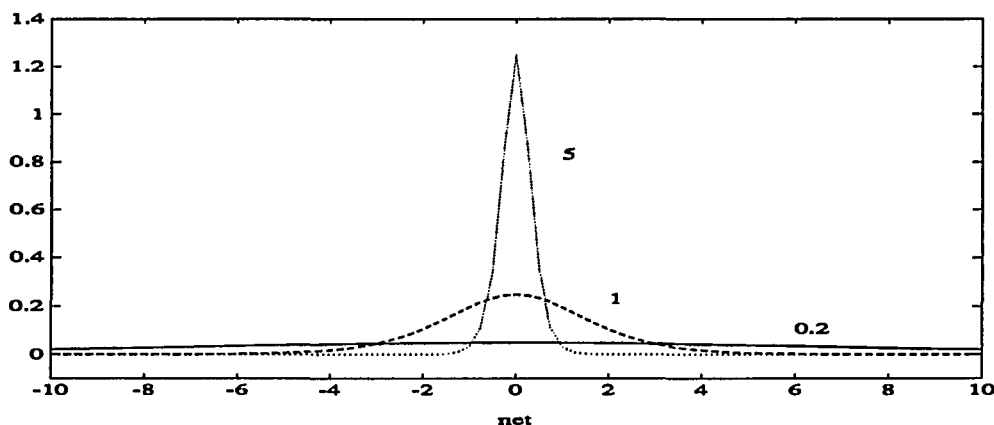


Figure 2.2: Derivative of the Sigmoid Function for Three Values of θ .

The constant, θ , in equation (2.3), is important in determining the slope of the sigmoid. It also plays a major role in training, as shown in equations (2.9) and (2.10). Izui and Pentland [19] and Rezgui and Tepedelenlioglu [34] show the significance of θ during training, indicating it is one of the most important determinants of learning rate, although Caudill states that the actual value is unimportant [5]. It is shown that the number of iterations to convergence is proportional to $1/\theta$ for GDR without momentum, and $1/\sqrt{\theta}$ for GDR with momentum [33], indicating

that a high value of θ (steep activation function slope) is desirable. Rezgui and associate [34] use a non-continuously differentiable activation function, violating a chief tenet of the GDR, to show that too steep a slope leads to saturation of the neurode and impedes or prevents training. This finding can be applied to sigmoidal AFs, too, and is particularly true if weights are initially so large (∞) or small ($-\infty$) that the value of “net” falls outside the dynamic portion of the AF.

None of the researchers referenced in section 2.2.3 give classification capabilities of networks trained with different parameters and empirical input data.

2.2.4 BPN Performance

Besides the topological and parameter considerations discussed above, there are other characteristics of BPNs that affect their performance. These may be broken down into training and classification considerations.

2.2.4.1 Training

The goal of training a BPN is to adjust neurode interconnection weights in the most efficient manner possible in order that the network be able to correctly classify known inputs (patterns that are the same, or very similar to, the training patterns) and unknown inputs (patterns with noise or unfamiliar features). It is usually desired that this training be accomplished in a timely fashion, particularly for “on-line” or adaptive applications. Since it is unrealistic to expect the LMS error to reduce to 0, thresholds are typically set that indicate an appropriate place to stop.

One threshold already introduced in section 2.2.1 is ϵ , the minimum error, where $\epsilon > 0$. This is necessary since it would take an infinite amount of time for equation (2.4) to reach 0. Once ϵ is set, the BPN is trained until the LMS error reaches this value. Since ϵ is a *mean* of all the pattern errors, statistically it should not matter how many patterns are used to train; the Gaussian distribution of error should be the same. In practical terms, though, it may be possible that a given set of patterns used to train the BPN may contain one or more pattern(s) whose *individual* error(s) is/are high enough that the network will not be able to recognize

the pattern(s) during classification, based on the criteria used (see the next section for more detail). For this reason, the individual pattern errors are sometimes used to decide when to stop training.

Choosing the value of ϵ can be challenging. As $\epsilon \rightarrow 0$ the longer it will take to train the network [5]. There is also the possibility that the smaller the value of ϵ , the less likely the network will be able to generalize and correctly classify unfamiliar patterns [6], i.e. the network will have become specialized for recognizing only those patterns that are very similar to ones already seen. On the other hand, the higher the value, the less likely the network will be able to recognize familiar patterns. Remember that the minimized LMS Error is a *mean*: therefore, even though the mean error for the entire training set may be low, one or more patterns may have large errors that will render them unclassifiable by the network.

Another threshold normally chosen is the maximum number of training iterations, T . T represents the number of times the BPN is to present all the patterns to the network, calculate the LMS error, and adjust the network weights, before finally giving up. If the number of iterations reaches T before the LMS error reaches ϵ , the BPN stops training. It is necessary to choose a T because some networks simply will not converge on a solution and will continue to iterate the BPN algorithms until the computing machine is unplugged. Sometimes the LMS error is very close to ϵ when T is reached; the network is just unable to take the one small step forward to ϵ in the T allocated. In these cases, the network may still be valid.

It would also be important to discover if networks that take a long time to converge on a solution are better or worse at classifying new patterns than networks that converge in shorter periods of time. In other words, is it really advantageous to speed-up the GDR, or will doing this degrade classification performance?

The number of patterns to use to train a BPN is the next consideration. Qualitatively speaking, enough patterns should be used so that the network can find the important features for each pattern class. The more patterns presented, the more likely the network will be able to determine these features. Adding more patterns, however, greatly increases training time. Gallinari *et al* give the most

stringent criteria for selecting the number of training patterns [9]. They say that, for input patterns of dimension n , the number of patterns, N , is always

$$n < N$$

and that this generally stands true in classification tasks. The literature is fairly mixed on this issue. Some researchers use $n \ll N$, while others use $n \gg N$. It is always the case, though, that

$$m \leq N$$

where m is the output layer size.

Finally, it is important to choose target values that the network can realistically achieve. While setting the target values to binary values (“0” and “1”) seems appropriate, in practice it would take an infinite amount of time for the sigmoid AF to reach these values. For this reason, the target outputs are usually selected as “0.9” for “on” and “0.1” for “off” [5]. It is also generally practiced that only one output neurode at a time be “on” to represent any one class.

2.2.4.2 Classification

Criteria for classification is another important network consideration. This may simply be stated as the value an output neurode will acquire based on the value at which it actually arrives. For example, after a network is trained, its weights have been adjusted to give the best possible output values (hopefully) based on the training patterns presented. Then the trained network is used for classification of known or unknown patterns. If a neurode now outputs a “0.7”, is this value good enough to decide the neurode is “on”? Classical statistical pattern recognition would say that any output < 0.5 belongs to class 1, and any output > 0.5 belongs to class 2. For BPNs, a lower and upper bound, τ_l and τ_u , respectively, may be chosen that represent confidence of selecting a neurode state based on the actual neurode output. The least stringent criteria would be for $\tau_l = \tau_u = 0.5$, while $\tau_l = 0.1$ and $\tau_u = 0.9$ would be the most stringent (for the case where “0.1” is “off” and “0.9” is “on”).

One other consideration is what to do during classification when more than one output neurode is “on” (in the case where only one output neurode may be active per pattern). Classical statistical pattern recognition would find the least costly prediction and assign the pattern to the appropriate class. This may lead to a high degree of misclassifications. BPN classification would most likely label these patterns as “unclassified”, which may be more desirable than misclassification, particularly in cases where misclassification can be costly (i.e. in the operating room).

None of the researchers referenced in section 2.2.4 gives classification capabilities of networks trained with different BPN performance criteria and empirical input data.

2.2.5 *A Final Consideration*

One final consideration for BPNs is what to do when training is being attempted simultaneously on two different types of input data. This would be particularly appropriate if, for example, it was desired that EEG *and* EKG data be used together to determine depth of anesthesia. Should the data be presented together as one input vector, or should some other technique be employed? For example, would two subnetworks, each trained on the different data types, connected to a final network that performed the actual classification be appropriate? None of the referenced research attempted to use more than one type of data for training and testing BPNs.

3.0 RESEARCH METHODOLOGY AND IMPLEMENTATION

This Thesis research required acquiring and preprocessing EEG data, writing and implementing BPN simulation programs for over 2,000 networks, testing trained networks for classification capabilities, recording and analyzing data, and comparing results with statistical paradigms. All computer programs used for this research were written in "C", and compiled and run on a Convex C-240, VAX, and SUN workstation.

3.1 EEG Data Acquisition and Preprocessing

3.1.1 *Data Acquisition*

EEG data used for this research was obtained from ten canine subjects under various stages of anesthesia. (This data was collected by members of the Department of Anesthesia, Arizona Health Sciences Center, prior to the inception of this Thesis research.) Canines were used because of the relative convenience of recording large amounts of data at various levels of anesthetic concentration. The Department of Anesthesia at the Arizona Health Sciences Center has used these dogs previously for anesthesia research and has demonstrated similarities between canine and human EEG response to anesthesia [14].

The canine subjects were prepared as follows (references in the next two paragraphs are to researchers who use similar techniques). Oxygen, nitrous oxide and isoflurane were first administered to ten greyhounds. N₂O was ceased after intubation, and an IV line was started to maintain fluids and drugs. Electrocardiogram, arterial blood pressure and end tidal volume were monitored and recorded at set intervals [2, 18, 23, 39]. Phenylephrine was used when necessary to keep blood pressure within 20% of baseline. Mechanical ventilation was used to keep end tidal

CO₂ between 35 and 40 torr [39]. The end tidal isoflurane (anesthetic) concentration was continuously monitored and adjusted using infrared spectroscopy [2]. Pancuronium was used for neuromuscular blockade.

EEG data was collected using electrodes placed subdermally in the frontal and occipital lobe regions of both the right and left hemispheres. The isoflurane concentrations were set at 0.7%, 1.4% and 2.1% for 15 minutes each in order to obtain three stable levels of anesthesia. These levels correspond to the stages of light, medium and deep anesthesia for dogs (referenced from here on as MAC 0.5, MAC 1.0, and MAC 1.5; MAC stands for minimum alveolar concentration; MAC 1.0 is the minimum alveolar concentration of an anesthetic drug required to render 50% of a given species immobile to a noxious stimulus). This provided thirty different cases to study (10 dogs \times 3 levels/dog).

A Diatec Lifescan EEG machine was used to monitor the subjects. EEG was stored on FM tape using a Hewlett Packard 3964A FM recorder [2, 18, 39]. The recorded EEG data was later visually monitored, and artifact-free continuous periods of 20-seconds for each MAC level were selected for further study [2, 23]. A relatively long period of time is needed in order to be sure of a steady-state patient status. The 20-second periods were digitized at 200 samples/sec [12, 18, 23, 39], well above the Nyquist criteria, with 12-bit resolution on an IBM AT using ASYST© software, providing 4,000 usable samples. These samples were stored on magnetic disk and later ported to the computer environments as needed. Figure 3.1 shows this sampled data for one subject at each MAC level.

3.1.2 *Front-end Processing*

Several front-end processing tasks were required in order to put the digitized EEG data into a usable format for the NN. These tasks are discussed in this section. As in the previous section, references in the next subsections are to researchers who use similar techniques.

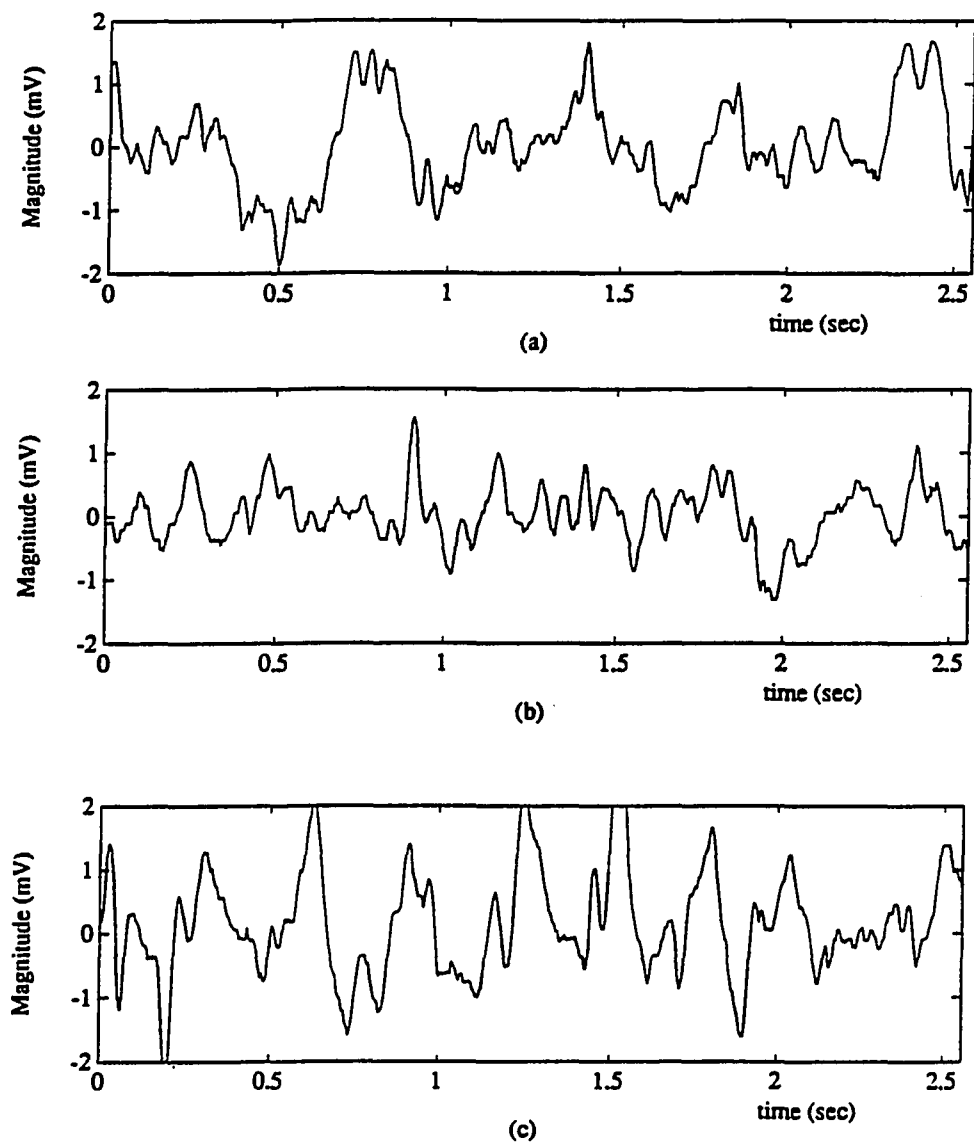


Figure 3.1: Sampled EEG for One Subject at Each MAC Level. (a) MAC 0.5. (b) MAC 1.0. (c) MAC 1.5. Sampling Frequency is 200 Hz.

3.1.2.1 *Discrete Fourier Transform*

As previously expressed in section 1.1.3, the power spectrum is believed to provide the best information about the EEG waveform. Therefore, the Fourier Transform of the data is needed in order to transform the time-domain data to the frequency-domain. For discrete data, such as that provided by the sampled EEG, the Discrete Fourier Transform (DFT) is used. The DFT output is a set of real and imaginary components for each frequency point.

A DFT program that was able to handle the data input and output formats was unavailable. This necessitated writing a DFT program from scratch. The Cooley-Tukey algorithm [31, 41] was used in order to speed-up the transformation. The 4,000 EEG data samples were first loaded and then divided into 7 epochs of 512 samples each (2.56 seconds per epoch) [12, 18, 23, 39]. A separate 512-point DFT was performed on each epoch, providing 0.39 Hz (200 Hz/512 samples) of frequency resolution [31].

Each of the seven transformed epochs had 512 sets of real and imaginary components for each frequency point. Only the first 60 sets of frequency-domain points (0.39–23.4 Hz) from each epoch were saved for further processing since the useful range of the canine EEG ends at roughly 20 Hz (see Figure 3.2). Finally, every-other sample point was discarded in order to decrease the NN input pattern size to a level that would allow faster computation while still retaining enough data resolution (raw data was checked to ensure that this “subsampling” of data did not produce aliasing). This left an input vector size of 30.

3.1.2.2 *DFT Power Spectrum and Phase*

As stated in the previous section, the DFT provides the real, R , and imaginary, I , parts of the frequency domain signal. The power, P , is found by taking

$$P = R^2 + I^2.$$

A program was written to calculate P for the set of thirty $[R, I]$ components in each of the 7 epochs calculated by the DFT program. P was in the range $[0 \dots \infty]$.

Figure 3.2 shows the EEG power spectrum for a subject at each of the three MAC levels.

Power spectrum is typically all the information used from the DFT. However, the phase, Φ , can also be calculated as

$$\Phi = \tan^{-1} \frac{I}{R}.$$

Oppenheim observed that many of the important features of a Fourier series are preserved if only phase is retained, while the same is not true in general for magnitude (power) [30]. While phase is used by some researchers in image processing [8], none of the literature on EEG processing suggests using phase as an alternative. This thesis research incorporated phase data in addition to power spectrum data in order to discover which was most useful. Another program was written that calculated Φ for the set of thirty $[R, I]$ components in each of the 7 epochs. The calculated Φ were in the range $[-\pi \dots \pi]$. Any $\Phi < 0$ were then made positive by adding 2π . This put Φ in the range $[0 \dots 2\pi]$. Figure 3.3 shows phase versus frequency for a subject at each of the three MAC levels.

Comparing Figures 3.2 and 3.3, it can be seen that each signal is extremely different in character. The magnitude range of the phase signal is much less than that of the power spectrum, and the relative magnitudes between phase peaks and valleys is more consistent. This could lead to different training and classification performance for the BPNs, and was considered an important characteristic to research in order to find an optimal NN classifier.

3.1.2.3 Data Smoothing and Normalization

In order to lessen the effects of transient signals and abnormal magnitudes, the 7 epochs of power and phase data were averaged [12, 38]. This was accomplished by a separate program that calculated the average power, \bar{P} , and the average phase,

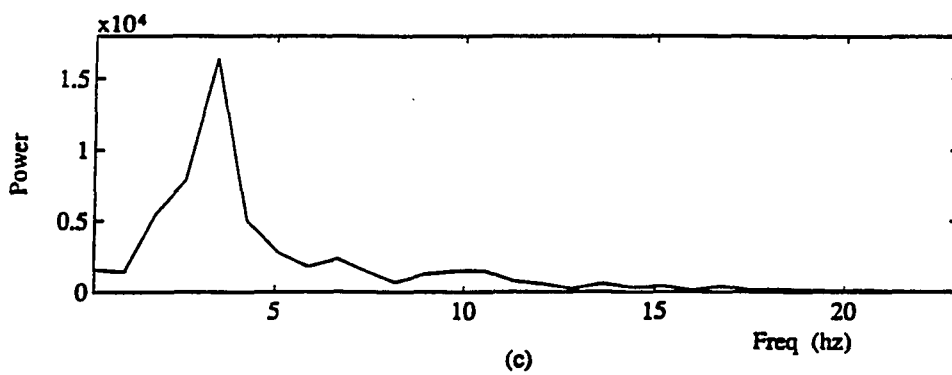
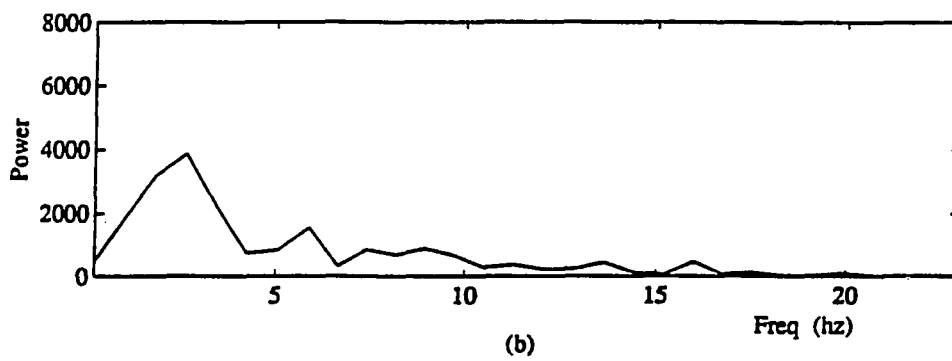
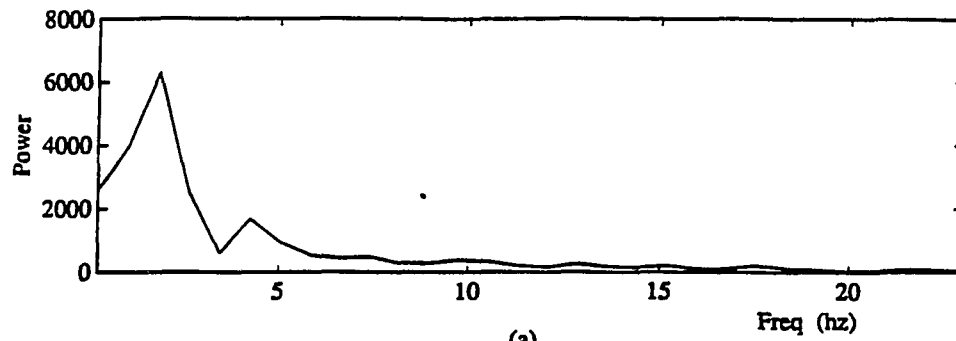


Figure 3.2: EEG Power Spectrum. (a) MAC 0.5. (b) MAC 1.0. (c) MAC 1.5 (note the change in scale).

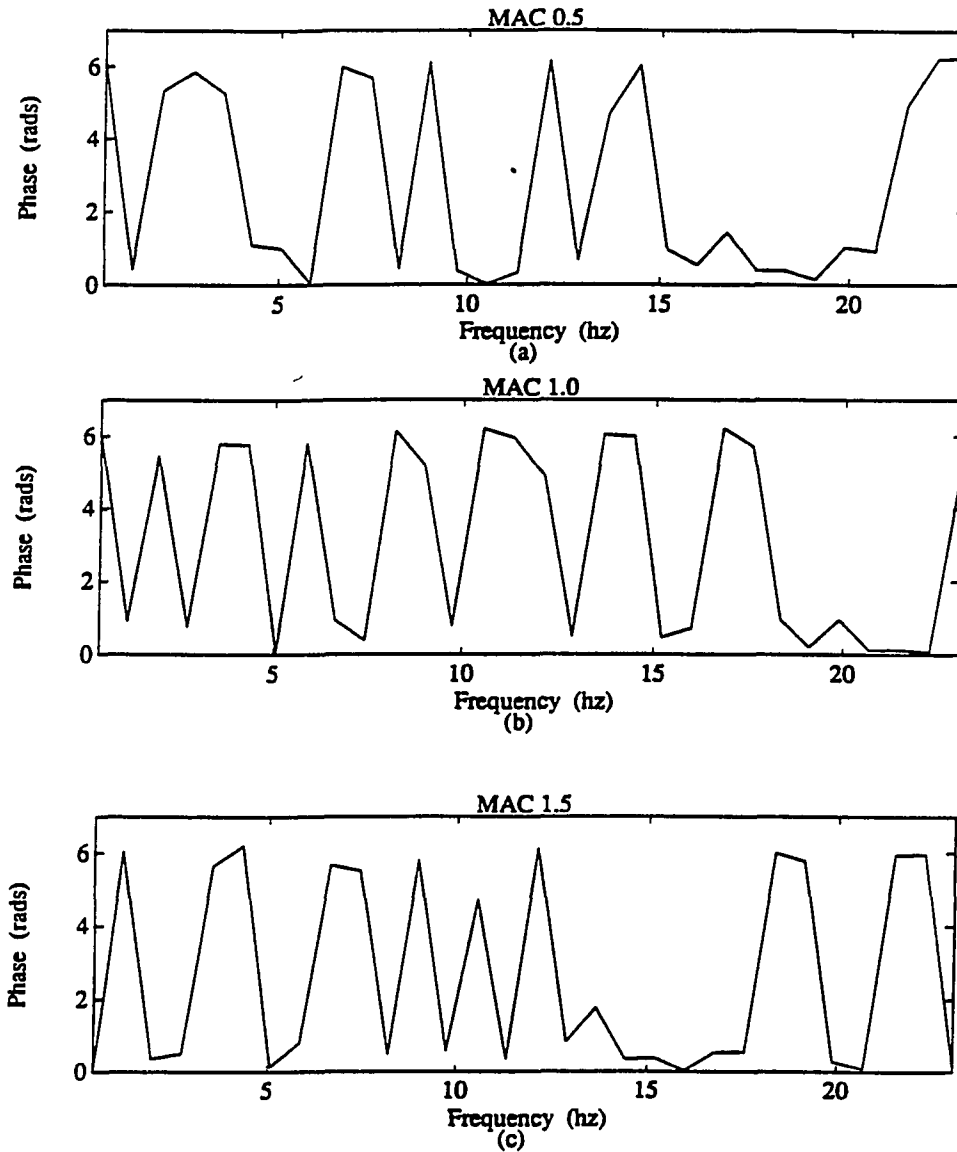


Figure 3.3: EEG Phase vs. Frequency. (a) MAC 0.5. (b) MAC 1.0. (c) MAC 1.5.

$\bar{\Phi}$ as follows

$$\bar{P}_j = \frac{1}{7} \sum_{i=1}^7 P_{ji}$$

$$\bar{\Phi}_j = \frac{1}{7} \sum_{i=1}^7 \Phi_{ji}$$

where j is the vector component number and $1 \leq j \leq 30$. Once the averages were found, the data were normalized in the range $[0 \dots 1]$ and written to separate power and phase data files and a data file containing both types of data combined.

3.1.2.4 BPN Input File Formation

During training, the BPN loads all the learning data at once. The network requires that each input pattern vector have an associated target output vector. Therefore, a single file containing all the training vector pairs is needed. A *BPN input file formation* program was written in order to accomplish this. This program prompts the user for the number of input patterns to use at each MAC level and then asks for the file name of the smoothed, normalized data (see previous section). The program then loads each pattern vector, appends a target vector, and writes the vector pairs sequentially to the BPN input file. The target vectors were specified as [001], [010], and [100] for MAC 0.5, 1.0, and 1.5, respectively. The program also wrote the number of vectors, the input vector size and the output vector size to the BPN input file.

3.2 BPN Training Simulations

Since physical Backpropagation Networks are not available, programs which simulate the behavior of BPNs were used. These simulations were used to build over 2,000 networks. Networks were trained using different parameters, topologies, and performance criteria (section 2.2), and different training sets. Performance data, including iterations to network convergence and final LMS Error, from these networks were then recorded and evaluated.

The main goal of testing BPN training capabilities was to discover which parameters, topologies, criteria, and data types have the greatest influence on

training time. Two secondary goals were to (1) attempt to address ambiguities in the literature concerning training, and (2) find optimum values to improve training capabilities.

3.2.1 *Simulation Programs*

The principal BPN simulation utilized for this research was a modification of the C language program presented by Pao [32, Appendix A]. His version contained two critical errors that needed to be corrected. First, the manner in which LMS error was calculated was flawed: Pao's version began by summing all the results of $(t_{pj} - o_{pj})$ (see equation 2.4) and *then* squaring the result. Second, Pao's version used an erroneous algorithm for adding momentum to the weight update equation. These two errors were corrected. The program was also modified to allow batch processing and more flexibility in the input files. The program was compiled and run under VAX/VMS, and UNIX on both a Convex C-240 mini-super and SUN workstation.

3.2.2 *Network Initialization*

EEG data files from the 10 canines were divided into two groups of five canines each. The groups were designated as GROUP1 and GROUP2. Each group contained power spectrum, phase, and combined data types for each dog. Each dog had three MAC levels for each data type, providing 15 input patterns per data type (5 dogs \times 3 levels) per dog, as shown in Figure 3.4. The program described in section 3.1.3.4 formed the BPN input files from these groups. Each group had three different input files for each of the three data types: power, phase, and combined. The BPN simulations were trained separately against each group and each data type. Fifteen input patterns is considered a relatively small training set. Ideally, this number would be much larger. The size of the training sets, however, was limited by the amount of data that was collected previously.

In batch mode, a separate "run-time specification" file provided the name of the input file and BPN topological and parameter values. This precluded having to recompile the program for each new run. This file instructed the simulation on

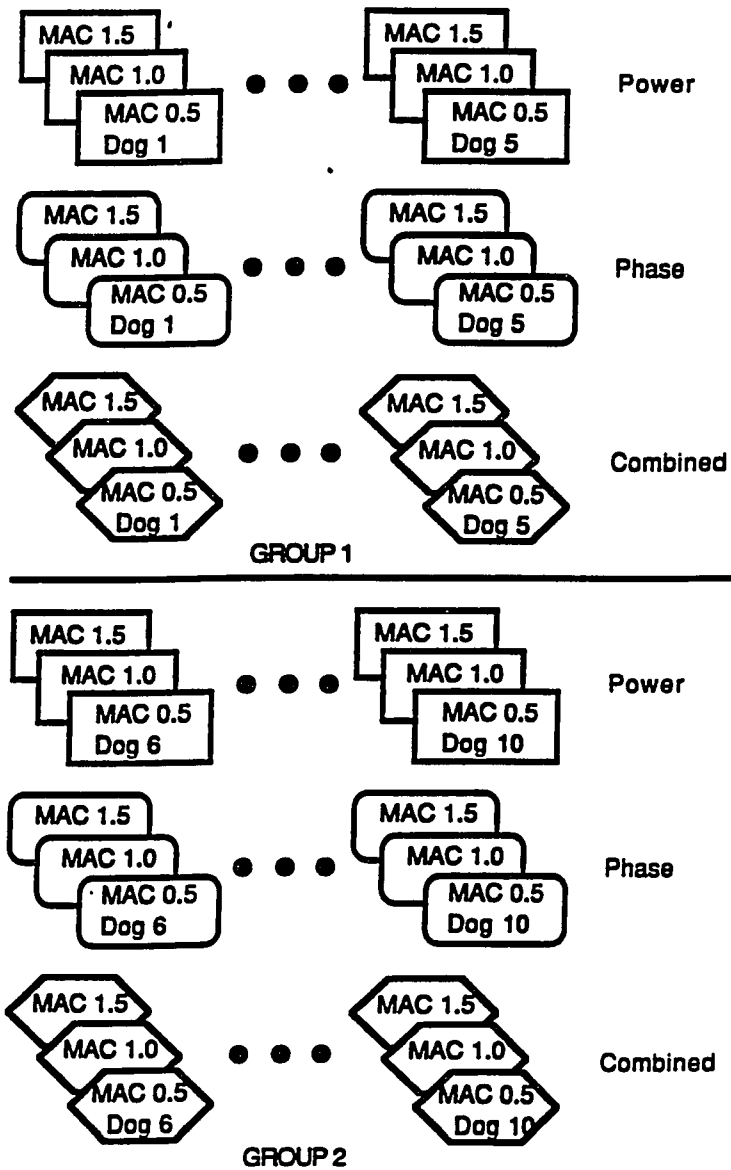


Figure 3.4: Input Patterns for Each Group: Power, Phase and Both.

how many middle layers to use, how many neurodes were in each middle layer, the value of θ for each middle layer, the maximum number of iterations, T , before aborting, and the LMS error criteria, ϵ (see sections 2.2.3, 2.2.4 and 2.2.4).

The simulation began by loading the input file for a given group and data type. As explained in section 3.1.2.4, this file specified the number of input patterns, and the input and output pattern sizes. Target patterns were equated as “0.0 = 0.1” and “1.0 = 0.9” (see section 2.2.4.1).

According to GDR theory, initial neurode weight values need to be unequal (section 2.2.1). To accomplish this, weights were initialized using a random number generator (RNG). The initial weight values also needed to be small enough that the value of “net” fell within the dynamic range of the AF (section 2.2.3.3). Therefore, the RNG output was constrained to the range $[-.25 \dots .25]$. Finally, networks with the same number of middle layers and middle layer neurodes needed to be initialized to the same weight values in order to accurately compare convergence times. This was accomplished by using the same RNG seed for each network, providing identical initial weights.

3.2.3 *Topology Variation Methods*

In order to test various theories about network topologies, and to actually find the optimum topology (section 2.2.2), it was required that BPN simulations be trained against a variety of architectures. To accomplish this, methods were used to alter the number of middle layer neurodes and the number of middle layers. For networks with one middle layer, the number of middle layer neurodes, M , was varied using $M = (0, 2, 4, 8, 12, 16)$. For networks with two middle layers, the number of first middle layer neurodes, M_1 , was varied using $M_1 = (2, 4, 8, 16)$, while also varying the number of second middle layer neurodes, M_2 , using $M_2 = (2, 4, 8, 16)$. Figure 3.5 shows BPN topologies for networks with one and two middle layers. The topologies were varied using the run-time specification file described above.

Two techniques were employed when attempting to train and classify on power spectrum and phase simultaneously (see section 2.2.5). The first technique

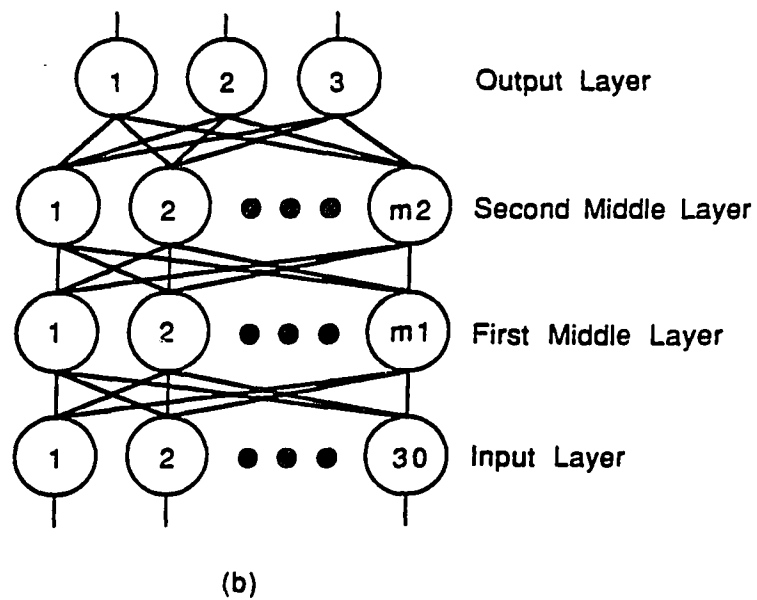
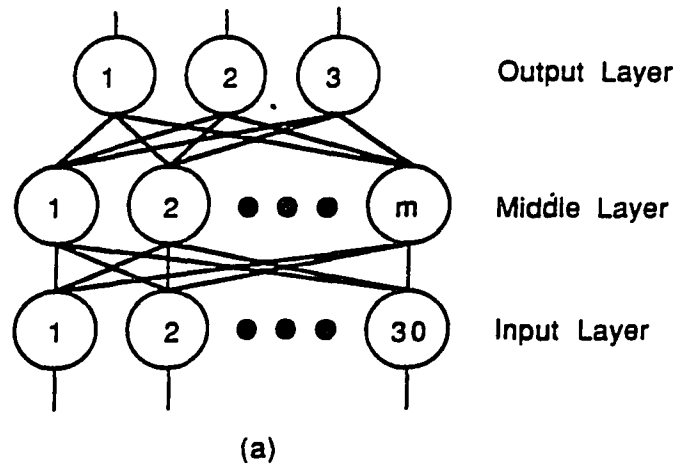


Figure 3.5: BPN Topologies. (a) Network with One Middle Layer. (b) Network with Two Middle Layers.

was the same as that described above. Power spectrum and phase vectors were presented to the network at the same time — as one vector with 60 features. The second technique trained two subnetworks that connected to a final network (see Figure 3.6). The power spectrum data was used to train one of the subnetworks while the phase data trained the other. Each subnetwork tried to form the appropriate target vector ([001], [010], or [100]). Outputs from the two subnetworks were then used as the input training vectors for the final network. The final network had an input layer of 6-neurodes, a variable middle layer size, and an output layer of 3-neurodes. Ideally, the final network would have input vectors [001001], [010010], and [100100], and corresponding target vectors [001], [010], and [100], respectively. (Recall that target vectors are equated as “0.0 = 0.1” and “1.0 = 0.9”.) Characteristics for the two subnetworks were chosen *a posteriori* from the networks that performed best for Phase and Power data separately.

3.2.4 Parameter Variation Methods

In order to test theories concerning training performance versus BPN parameters (section 2.2.3), several techniques were required to alter these values and test their effect on network training (and classification). The parameters η , α , and θ were varied over several values. For a given network topology and fixed θ , η was varied using $\eta = (.1, .3, .5, .7, .9)$ while simultaneously varying α using $\alpha = (.1, .3, .5, .7, .9)$. This provided 25 separate results for a given network topology and θ . The values of η and α were chosen to give a wide variation in the required range [0...1].

The value of θ for each layer was also varied using $\theta = (0.2, 1.0, 5.0)$. The θ 's for each layer were varied simultaneously using the specified values. This gave a total of 75 separate results for a given network topology. The θ values were chosen to reflect the most widely selected value ($\theta = 1$), and a range from one-fifth to five-times the “popular” value.

The values of η and α were varied by using nested **FOR** loops in the BPN simulation. The value of θ was varied using the run-time specification file described previously.

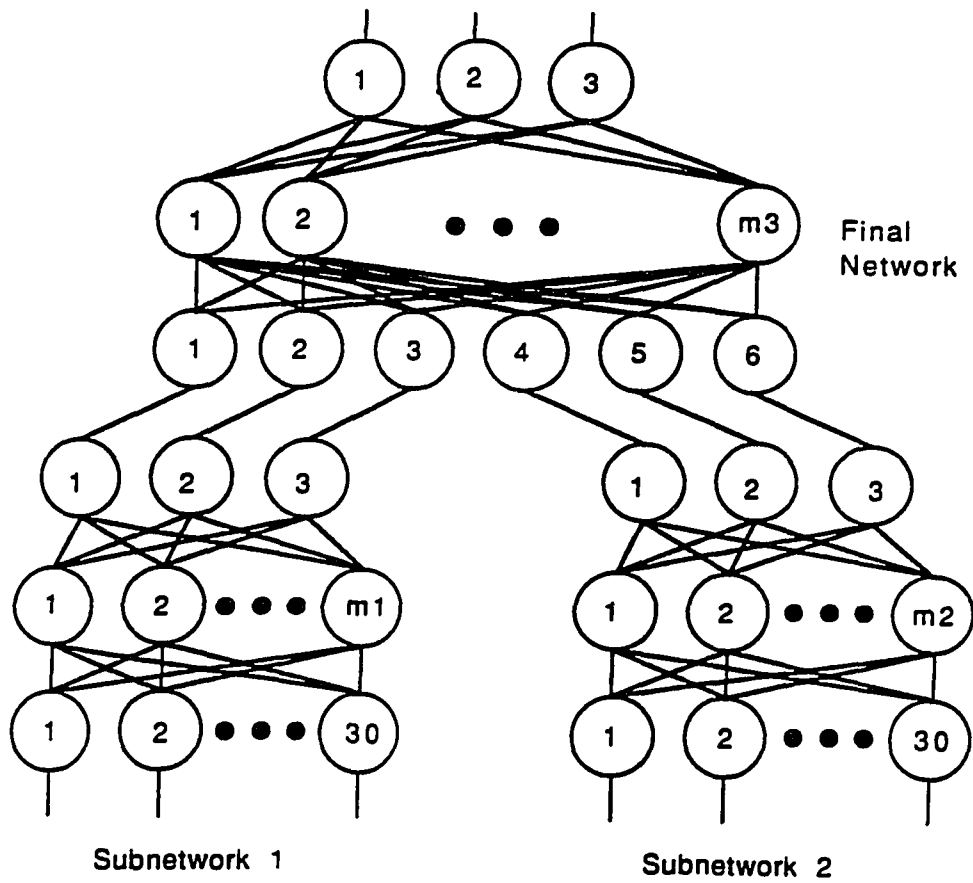


Figure 3.6: Multiple Network BPN for Two Different Input Types. Each subnetwork may have one or more middle layers. The final network was restricted to one middle layer.

3.2.5 *Performance Criteria Variation Methods*

The maximum number of iterations before aborting, T , was always set to 30,000 (section 2.2.4.1).

The LMS error criteria, ϵ , was set to .001 for most of the training (section 2.2.4.1). When certain networks showed interesting training and classification results, this value was varied using $\epsilon = (.001, .005, .025, .625)$. ϵ was varied using the run-time specification file described above.

3.2.6 *Training Performance Recording Methods*

Simulation results were recorded for all the variations in topology, parameters, and performance criteria.

During training, the values of the LMS error versus iteration number were occasionally recorded in a separate data file. This was done in order to discover if there were similarities in the way different networks minimized LMS error. This was normally only accomplished for networks with parameters that converged in fewer than a few thousand iterations. Otherwise the data file grew too large for the disk space available.

Once the network converged, or the iterations reached T , the network topology, parameters, performance criteria, number of iterations and final LMS error were written to a separate data file. These values were then entered into a spreadsheet for further evaluation.

The final values of the weights for specific neurodes could also be recorded in a separate data file. This was done occasionally in order to see how the first middle layer performed feature extraction on the input vectors. The values were graphed for magnitude versus frequency (feature), where large negative values signified the feature was unimportant in training, and large positive values signified the feature was important.

3.3 BPN Classification Simulations

A subset of the trained BPNs were used to test classification capabilities. To test classification capability, a network trained with GROUP1 data was tested with GROUP2 data, and vice versa (i.e. train on 15 patterns, classify on 15 new patterns). This criteria was much more strict than that used in most of the relevant literature. Typically, much larger training sets are used and either a subset of the training patterns or a much smaller new set of patterns is used to test classification capabilities. "Classification score" was recorded by counting the number of correctly classified patterns and dividing by the total number of patterns.

The main goal of testing the classification capabilities of the BPNs was to discover if the networks could efficiently classify the different types of EEG data. Secondary goals were to find what network topologies were best for specific types of data and to discover if methods used to improve training capabilities hindered or helped classification.

3.3.1 *Simulation Programs*

Simulation programs used to test classification ability were the same as those used for training, except that all the topology, parameter, and performance criteria considerations could be specified directly in the run-time specification file.

3.3.2 *Parameter Variation Methods*

The only parameter relevant to classification was the activation function constant, θ_C , since η and α only affect the weight update equations. Rezgui *et al* [34] discuss varying θ during training in order that "net" be in the dynamic range of the AF for large LMS errors, thereby increasing adaptation rate. θ would then be decreased as the LMS error decreased, thereby decreasing adaptation rate. Although the idea seems sound, it involves adding complexity to the GDR. However, increasing the value of θ during *classification* is a simple matter and could lead

to better accuracy. To accomplish this, θ_C was varied using $\theta_C = (.2\theta_T, \theta_T, 5\theta_T)$, where θ_T is the value of θ used during training.

3.3.3 Performance Criteria Variation Methods

An important consideration during classification is how to evaluate output values that are unequal to target values (section 2.2.4.2), or, more specifically, what is to be done with an output of [0.2 0.8 0.3]? Two different criteria were used. The first criteria stated that any output value, o_{pj} (see equation 2.4), $\leq .33$ was evaluated as “off” (0.1), any $o_{pj} \geq .67$ was evaluated as “on” (0.9), and $.33 < o_{pj} < .67$ was evaluated as “undecided” (0.5).

The second criteria was much simpler and stated that $o_{pj} < .5$ was “off” and $o_{pj} \geq .5$ was “on”.

The evaluated outputs were compared with the target outputs and correct classifications tallied. If more than one o_{pj} were “on”, or if all o_{pj} were “off”, the pattern was considered “unclassified”.

3.3.4 Classification Performance Recording Methods

Network topology, parameters and performance criteria were recorded to a separate data file, along with the percent correct classification and the LMS error between actual output and target output. Classification scores for individual input patterns were occasionally recorded in order to find trends for patterns which were classified/unclassified more often.

3.4 Statistical Analysis of EEG Data

In order to test the advantages of Neural Networks and to verify the findings of Gallinari *et al* [9], a statistical paradigm was needed to evaluate the same EEG data classified by the BPNs. This was accomplished with discriminant analysis using SPSS© statistical software [29]. The EEG input files were loaded into SPSS using the same groupings described in section 3.2.2. Separate discriminants were found for each group. The discriminants were then used to classify GROUP1 and GROUP2 data. Classification scores were recorded in separate data files. It

should be noted that discriminant analysis *always* classifies an input (“unclassified” patterns are not allowed), and this choice is based on a Bayesian classifier that gives the lowest cost for a given choice. This method can lead to misclassification of both training data and testing data. Results of the statistical analysis were compared with those from the BPN analysis.

4.0 RESULTS AND DISCUSSION

The results and discussion in this chapter correspond with the methodology and implementation sections in chapter 3. Comments are divided into Training and Classification subsections in order to better denote the differences in performance for network characteristics. These subsections are further divided into a “Results” section, where the empirical data is simply stated, and a “Discussion” section, where the results are compared and contrasted.

4.1 BPN Training Simulations

Over 2,000 networks were formed to collect the following data. This provided a considerable data base for researching BPN training characteristics and applications to EEG monitoring.

4.1.1 Results

4.1.1.1 Data Type — Power, Phase, Combined

Of all the characteristics used to train the BPNs, the data type was *the most important factor* to training time. Power spectrum data, with its large variations in amplitude, was the most difficult data type upon which to train. Training with Power data was also the most susceptible to changes in the other network parameters. The output neurodes were prone to saturation. In fact, roughly 50% of the networks formed were unable to converge on a solution in the 30,000 iterations allotted.

Phase was by far the most useful data type for training. Whereas Power data typically took thousands of iterations to train, Phase data could train in under 100 iterations. This fact led to some initial enthusiasm with respect to applying this discovery to EEG monitoring, which was later tempered by classification results

(see section 4.2). The Phase-trained networks rarely became saturated. In addition, training showed stable performance over a wide range in variations of other network characteristics; this performance could easily be modeled using regression analysis (see sections 4.1.1.6 and 4.1.1.7). When compared with Power-trained networks, Phase trained networks showed smoother training performance as LMS Error was minimized (see sections 4.1.1.3, Figures 4.2 and 4.4).

Combined data showed similar results to Phase data, although training time was usually greater than that for Phase alone. Like the Phase data, training showed stable performance for variations in other network characteristics, which could be modeled (see sections 4.1.1.6 and 4.1.1.7).

4.1.1.2 *Group1 vs. Group2*

The two groups showed similar results over variations in the network characteristics. The major difference was that Group2 trained 1.5- to 4.0-times faster than Group1 for all the data types.

4.1.1.3 *Middle Layers*

For Phase data, training performance was better for networks with no middle layer or one middle layer. Feature extraction was only slightly altered for networks with one or two middle layers where the first middle layer was the same size. Figure 4.1 shows two examples of feature extraction for networks trained with one middle layer (4.1 (a) & (d)) and then with two middle layers (4.1 (e) & (f)).

Adding a first or second middle layer always increased the iterations to convergence. A second middle layer made the training performance more inconsistent when other network parameters were changed. LMS error was larger at first and dropped at a slower rate for networks with two middle layers as compared to networks with one middle layer. Figure 4.2 demonstrates this for the sample networks used in Figure 4.1. Notice that in Figure 4.2 (e) & (f), for networks with two middle layers, LMS Error begins to drop significantly in the final half of training

iterations; LMS Error drops significantly in the first half of training iterations for networks with one middle layer, as shown in Figure 4.2 (a) & (c).

For Power data, networks with no middle layer rarely reached convergence. The number of iterations typically was greater than 20,000. If a middle layer was added, the networks showed more consistent and stable performance over a wider variety of network characteristics. Power-trained networks showed different feature extraction characteristics than Phase-trained networks: feature extraction varied more significantly at lower frequencies when a second middle layer was added, as shown for the examples in Figure 4.3 (a) and (b). Interestingly, feature extraction was only slightly altered (for neurode 1) if the first middle layers of two separate networks were different sizes, as shown in Figure 4.3 (b) & (d).

Adding a second middle layer usually improved the Power training performance significantly, as demonstrated for the sample networks in Figure 4.4 (a)–(d): iterations even dropped to under 1,000 for several of the networks, as demonstrated in Figure 4.4 (b). Note that although oscillations increased, iterations to convergence dropped. In fact, Figure 4.4(c) shows a network that was unable to converge with one middle layer (LMS Error remained consistent at .0125 through the 30,000th iteration, although only the first 3,000 iterations are shown), while Figure 4.4(d), with a second middle layer, could converge in under 3,000 iterations. Performance with a second middle layer was also much more stable over a wider variety of network training characteristics.

4.1.1.4 Middle Layer Size(s)

Training performance using Phase data was barely affected by the middle layer size of a one-middle layer network, particularly when η and α were in the middle of their ranges. As the layer size increased, there was usually a *slight* drop in the number of iterations to convergence. At some point, adding neurodes often increased the number of iterations, implying that there is an optimum middle layer size. This point varied with the values of θ_2 and θ_3 . The examples shown in Figure 4.5 (a)–(d) demonstrate the varied results for Phase-trained networks.

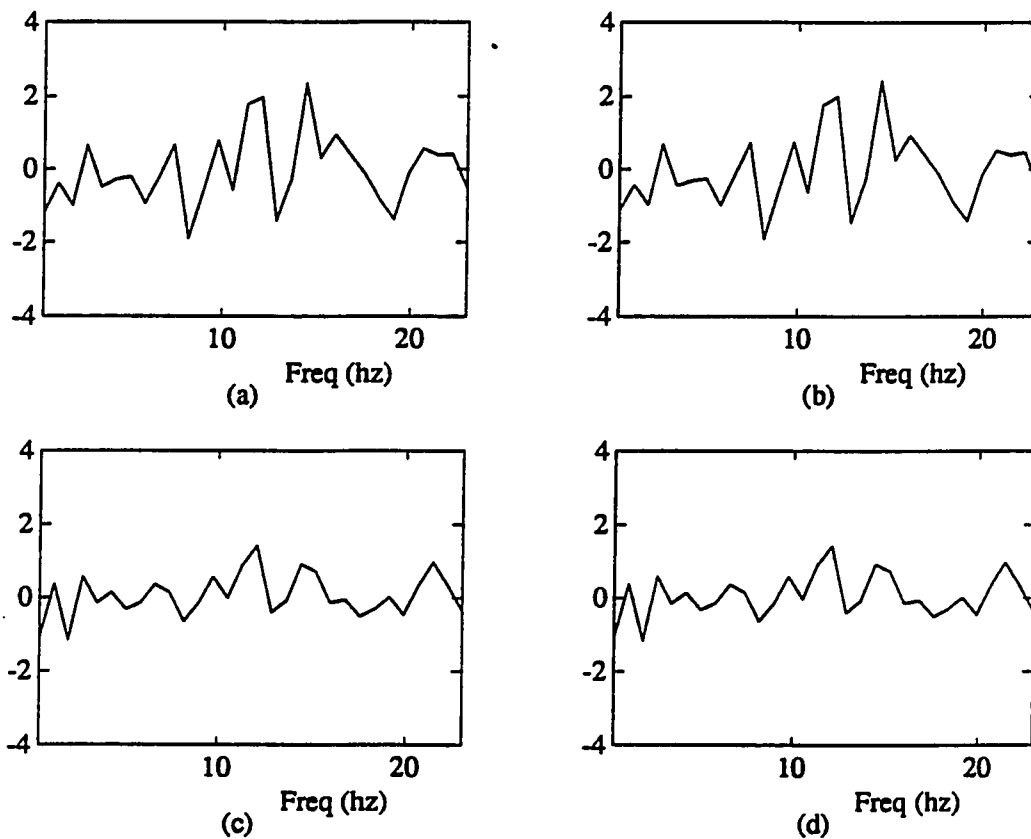


Figure 4.1: Weight Magnitude vs. Frequency for Neurode 1 of the First Middle Layer for Networks Trained with Phase Data. (a) One Middle Layer With 4 Neurodes, $\eta = \alpha = 0.5$. (b) One Middle Layer With 4 Neurodes, $\eta = \alpha = 0.1$. (c) One Middle Layer With 12 Neurodes, $\eta = \alpha = 0.5$. (d) One Middle Layer With 12 Neurodes, $\eta = \alpha = 0.1$.

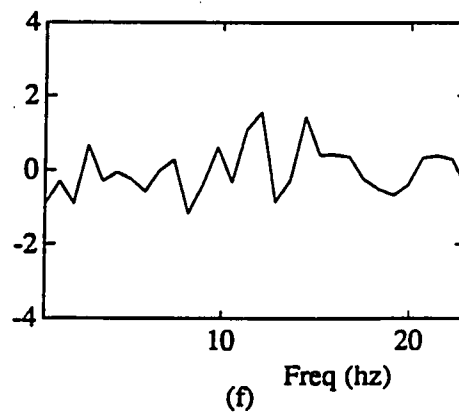
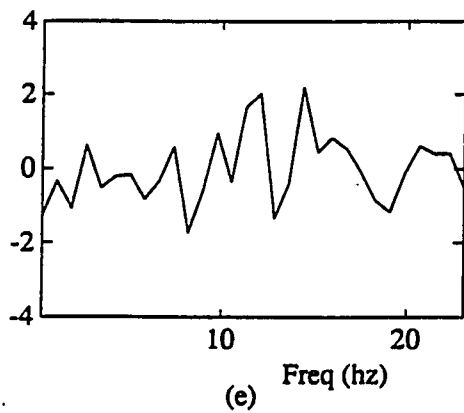


Figure 4.1 (continued) (e) Two Middle Layers With 4 Neurodes Each, $\eta = \alpha = 0.5$. (f) Two Middle Layers With 12 Neurodes Each, $\eta = \alpha = 0.1$.

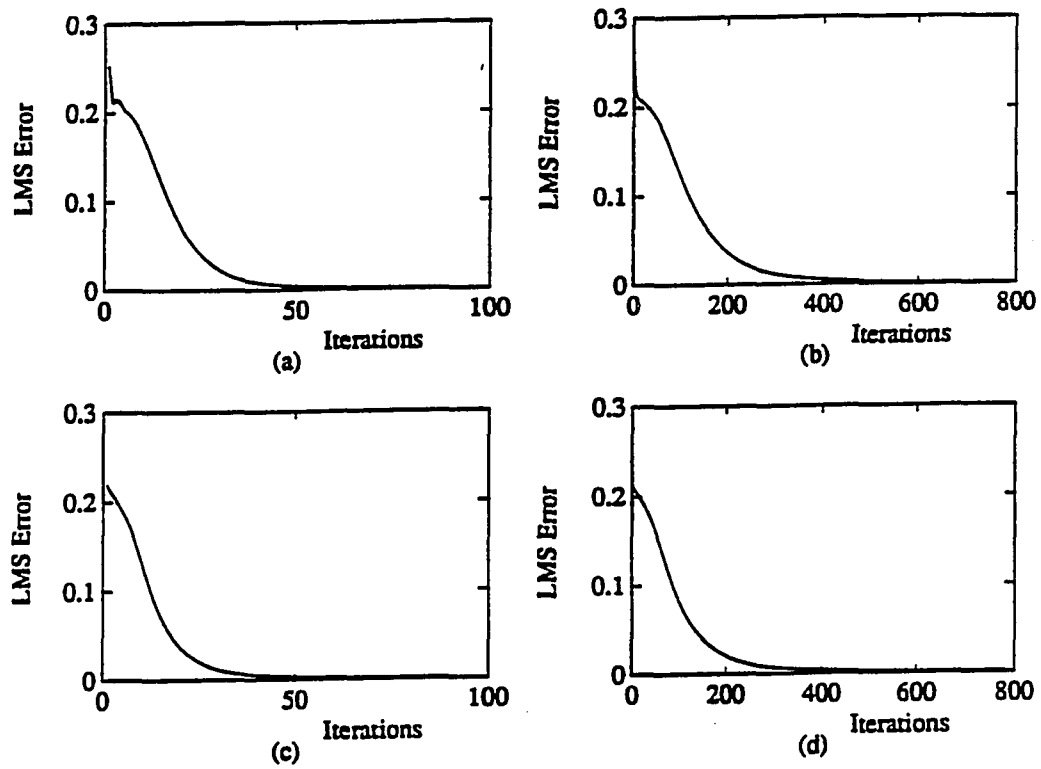


Figure 4.2: LMS Error vs. Iterations for Networks Trained with Phase Data. (a) One Middle Layer With 4 Neurodes, $\eta = \alpha = 0.5$. (b) One Middle Layer With 4 Neurodes, $\eta = \alpha = 0.1$. (c) One Middle Layer With 12 Neurodes, $\eta = \alpha = 0.5$. (d) One Middle Layer With 12 Neurodes, $\eta = \alpha = 0.1$. Note change in x-scale for (b) and (d).

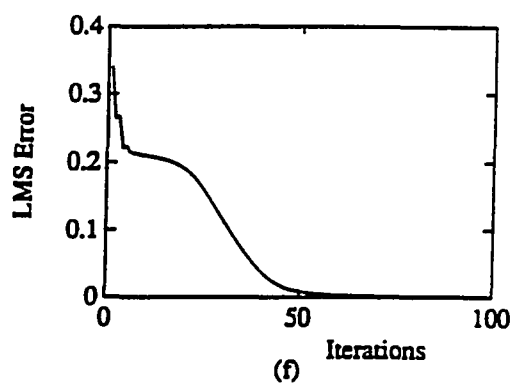
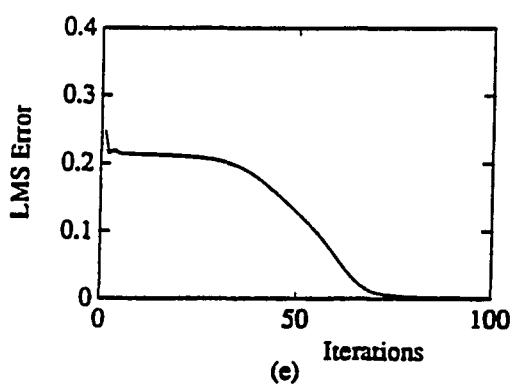


Figure 4.2 (continued) (e) Two Middle Layers With 4 Neurodes Each, $\eta = \alpha = 0.5$. (f) Two Middle Layers With 12 Neurodes Each, $\eta = \alpha = 0.1$.

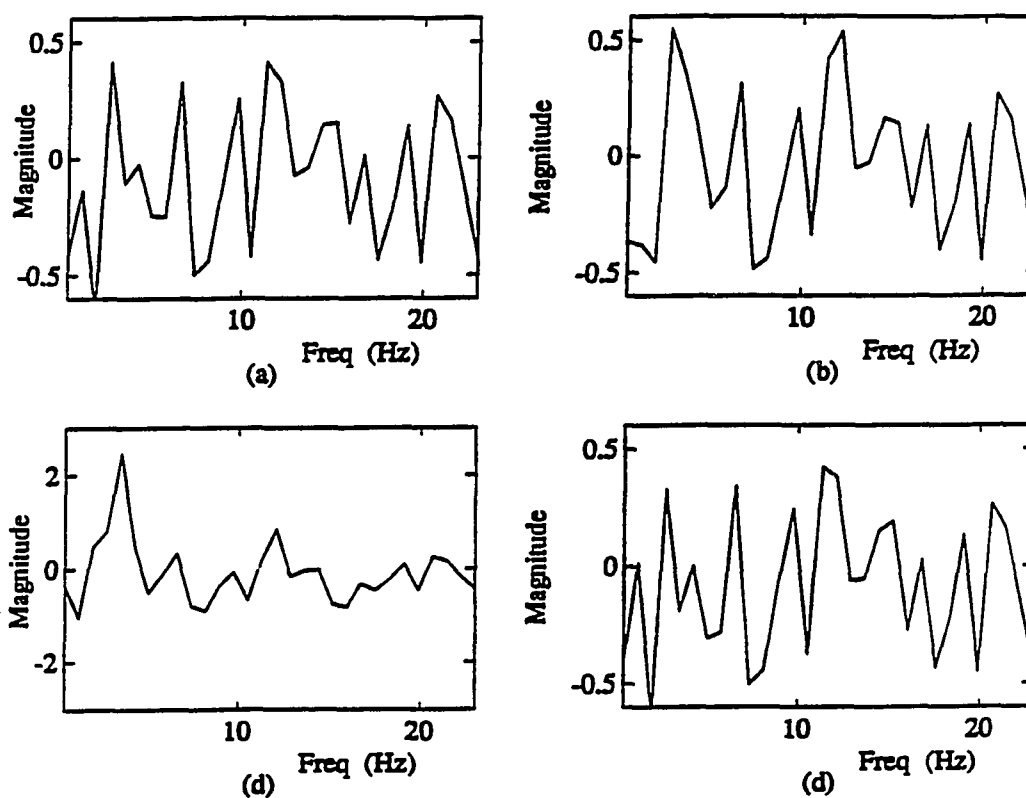


Figure 4.3: Weight Magnitude vs. Frequency for Neurode 1 of the First Middle Layer for Networks Trained with Power Data. (a) One middle layer, 4 neurodes. (b) Two middle layers, 4 neurodes each layer. (c) One middle layer, 12 neurodes. (d) Two middle layers, 12 neurodes each. For (a)–(d), $\eta = \alpha = 0.1$ and $\theta = 5.0$ for all layers. The network in (c) was the only BPN not to converge.

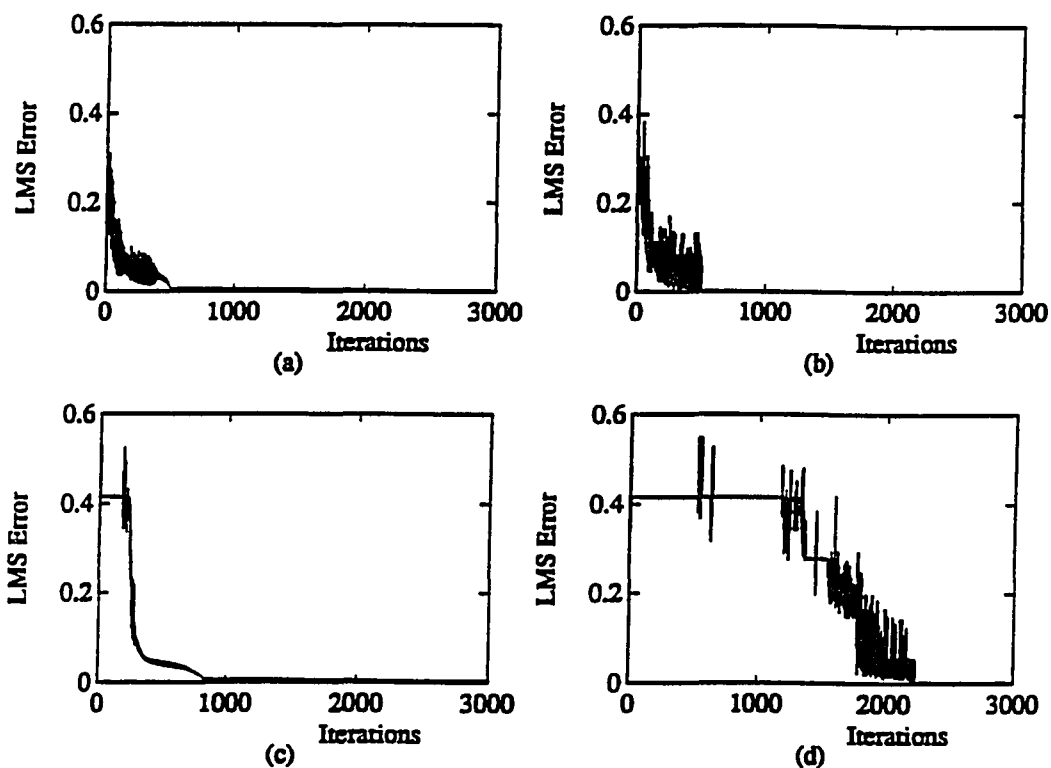


Figure 4.4: LMS Error vs. Iterations for Networks Trained with Power Data. (a) One middle layer, 4 neurodes. (b) Two middle layers, 4 neurodes each layer. (c) One middle layer, 12 neurodes. (d) Two middle layers, 12 neurodes each. For (a)-(d), $\eta = \alpha = 0.1$ and $\theta = 5.0$ for all layers. The network in (c) was the only BPN not to converge.

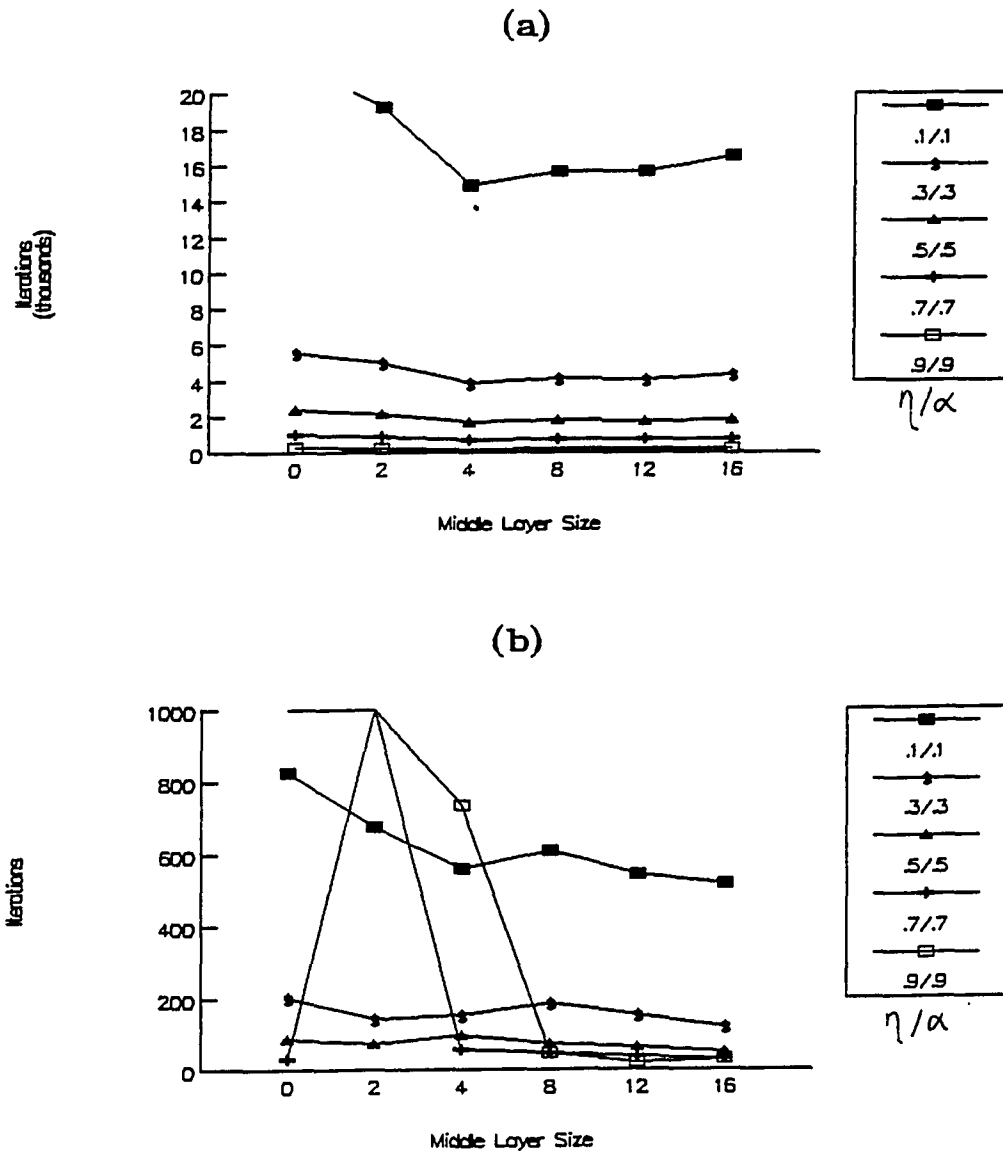


Figure 4.5: Iterations vs. Middle Layer Size for Phase-trained BPNs for Different Combinations of η and α . (a) $\theta_2 = \theta_3 = 0.2$. (b) $\theta_2 = \theta_3 = 1.0$. (Group1.) Legend to right indicates η/α .

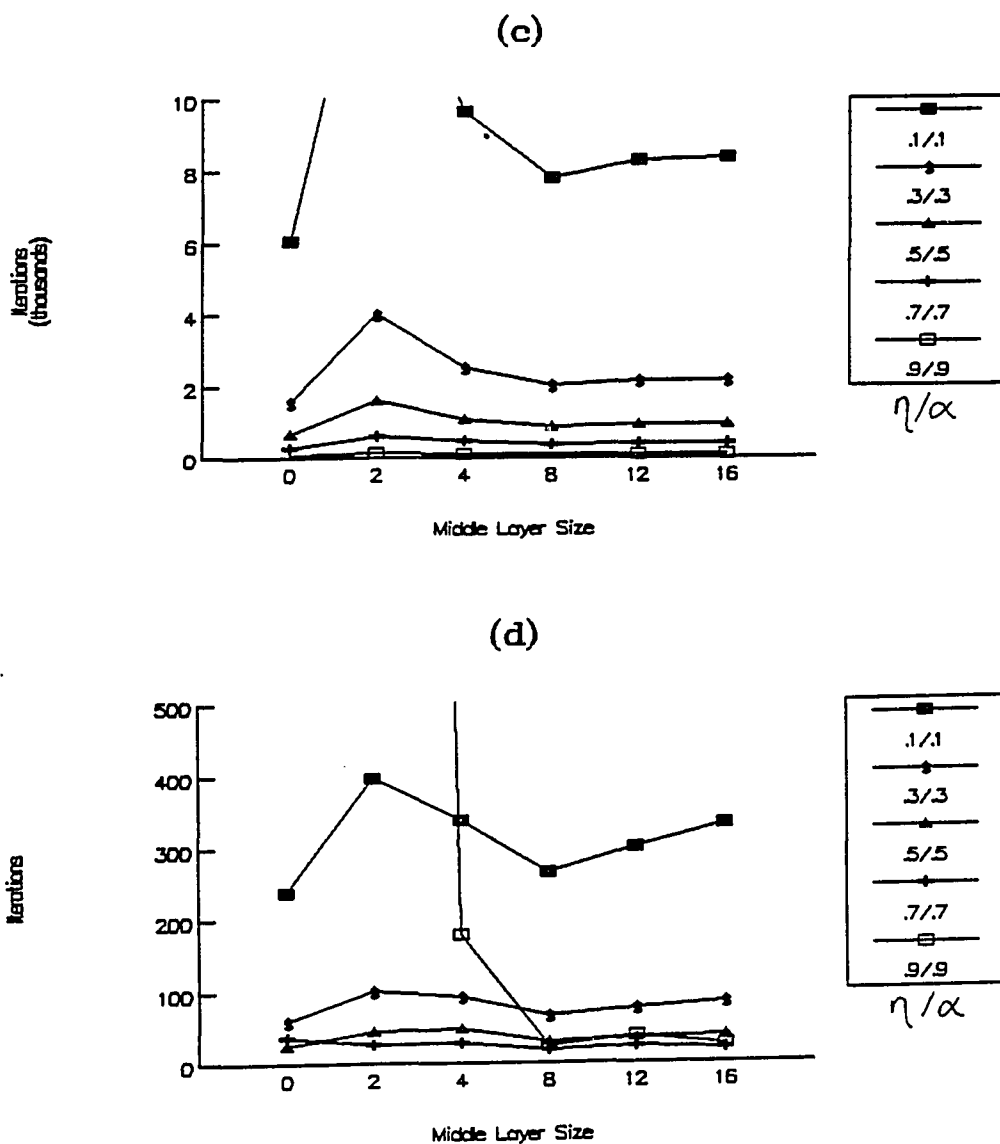


Figure 4.5 (continued) (c) $\theta_2 = \theta_3 = 0.2$. (d) $\theta_2 = \theta_3 = 1.0$. (Group2.)

For networks with two middle layers, the sizes were more important. Networks with smaller numbers of neurodes in both middle layers always took longer to train than those with larger sizes. Relative size between the two layers was also important. Networks with a larger first middle layer than second middle layer normally took longer to train than those with the reverse characteristics. A network with middle layers smaller than another network showed larger variations in relative weight magnitudes, as shown in the examples of Figure 4.1 (a) & (c) and (e) & (f), while a network with larger middle layers showed a slightly increased rate of drop from maximum LMS Error to a stable error, as shown in the examples of Figure 4.2 (a) & (c) and (e) & (f).

For Power data, middle layer size(s) greatly affected the training capability of the networks, although performance was extremely inconsistent. There was a general trend that, for networks with one middle layer, iterations to convergence dropped slightly as layer size increased. There was some point where iterations began to increase again as layer size increased, but this point fluctuated greatly with the values of θ_2 and θ_3 . For networks with two middle layers, performance became much more consistent. Networks with four neurodes in the first and second middle layers seemed to be optimum, as shown previously in Figure 4.4. Networks with larger first middle layers than second middle layers outperformed those with reverse characteristics (just the opposite from networks trained with phase data).

Networks trained with Combined data showed similar performance to those trained with Phase data. Iterations to convergence was only slightly affected by adding neurodes to the middle layer, as shown in the examples in Figure 4.6 (a)–(d). Feature extraction by networks trained on Combined data was examined in order to discover how the networks treated Phase and Power when they were presented together. As expected from the training results, weight magnitudes showed that Power data was much less important than Phase data in determining neurode activity. Figure 4.7 is an example of the differences in feature extraction for networks trained on Power or Phase data alone, and on Combined data. Figure 4.7 (a) shows the weight magnitudes for the first middle layer neurode for a network trained on Power data alone, while (b) shows the weight magnitudes for corresponding input

lines to the same neurode for a network trained on Combined data. The figure shows that, for combined data, net_j (equation 2.1) is approximately 0 for Power inputs since the peak weight magnitudes have been reduced to near-zero values. These results are contrasted with those in Figure 4.7 (c) and (d). Figure 4.7 (c) is for a network trained on Phase data alone, while (d) is for the corresponding input lines for a network trained on Combined data. The feature extraction is very different between the two networks, indicating that magnitude data has affected the results in the Combined data-trained network, but peak magnitudes have remained the same.

For all types of data, networks with two middle layer neurodes typically showed the slowest training performance: even networks with no middle layer performed better.

4.1.1.5 *Multiple Network BPN*

Networks with the best performance results on Power and Phase data separately were chosen *a posteriori* as the subnetworks for this BPN. For the Phase subnetwork, this was any network with one middle layer, $\eta = \alpha = 0.7$, and $\theta = 1.0$. For the Power subnetwork, this was a network with four neurodes in the first middle layer, four in the second middle layer, $\eta = \alpha = 0.1$, and $\theta = 5.0$. The final network showed good training performance over the full range of middle layer sizes; training was best when there was no middle layer. Although the total number of iterations (subnetwork 1 + subnetwork 2 + final network) was more than for any one individual network, training performance was much more consistent for the multiple network BPN, and the classification capabilities were greatly improved (see section 4.2.1.5).

4.1.1.6 *Learning Constant — η*

The learning constant had a fairly consistent effect on the training performance of the networks. That is, as η increased, iterations to convergence decreased.

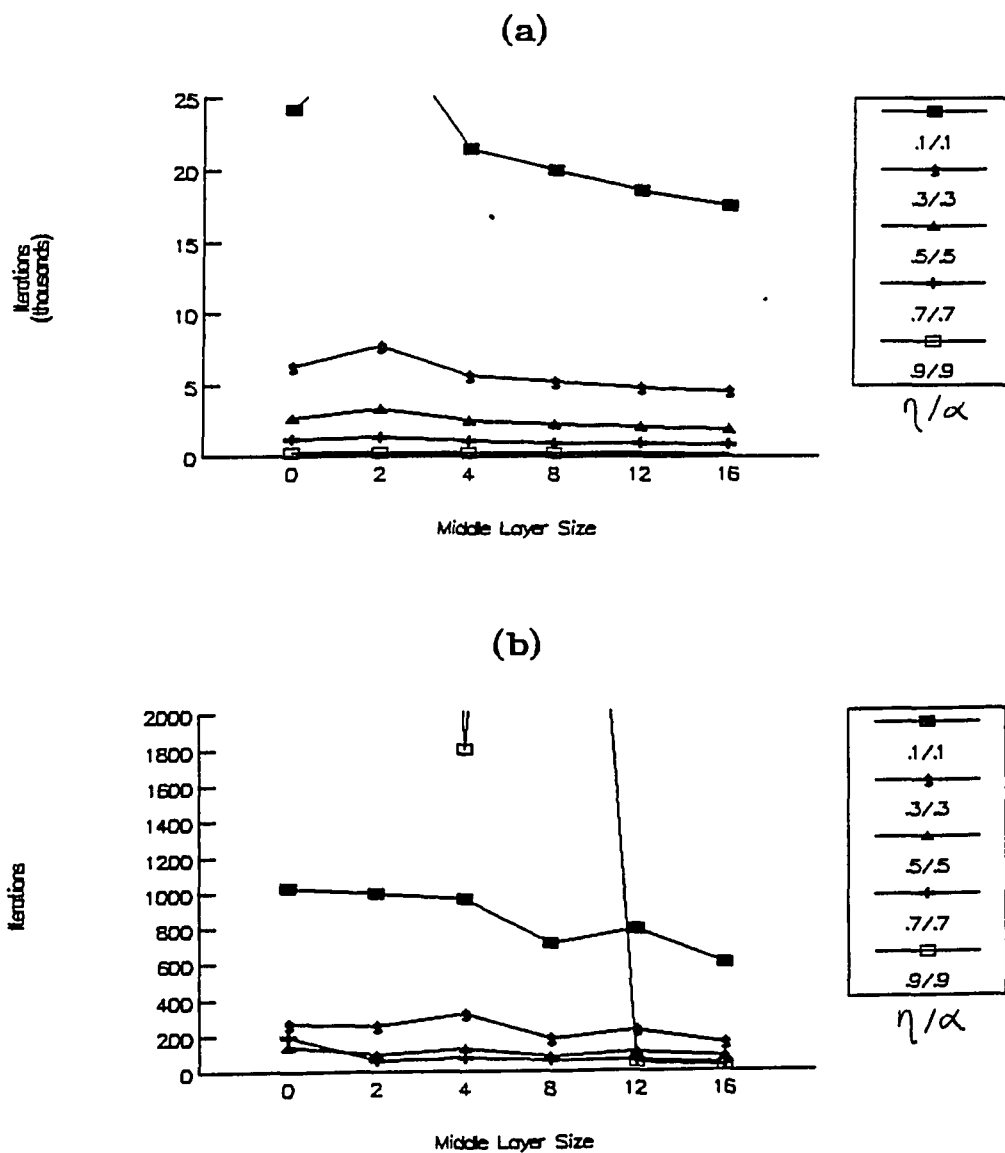


Figure 4.6: Iterations vs. Middle Layer Size for Combined Data-trained BPNs for Different Combinations of η and α . (a) $\theta_2 = \theta_3 = 0.2$. (b) $\theta_2 = \theta_3 = 1.0$. (Group1.)

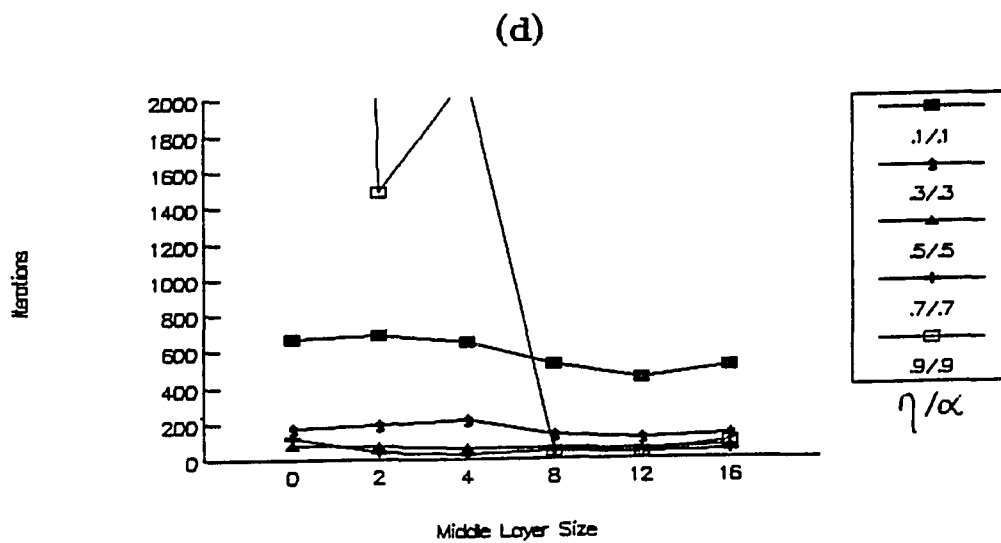
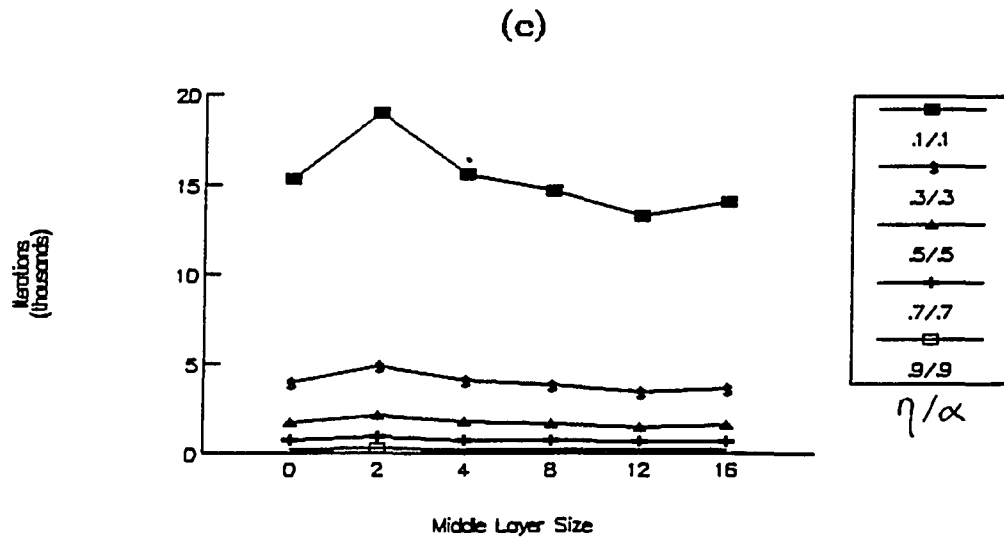


Figure 4.6 (continued) (c) $\theta_2 = \theta_3 = 0.2$. (d) $\theta_2 = \theta_3 = 1.0$. (Group2.)

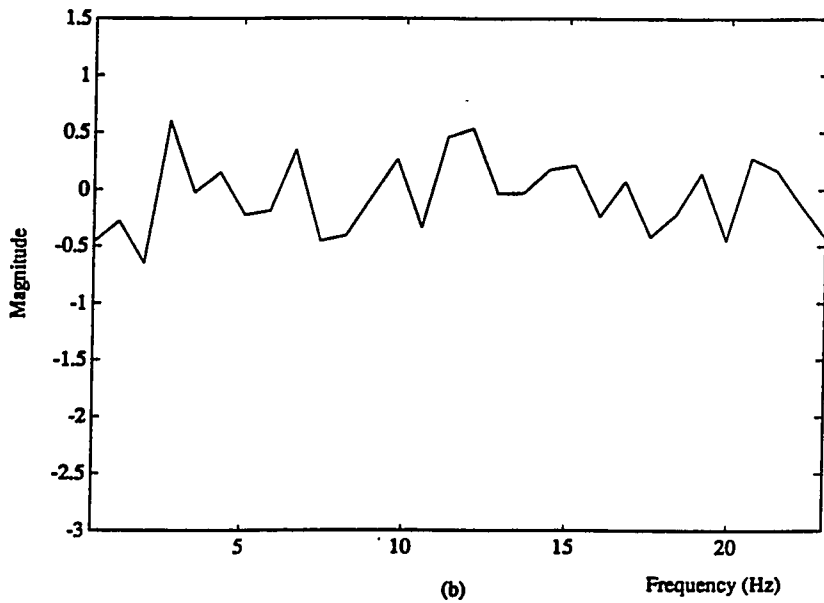
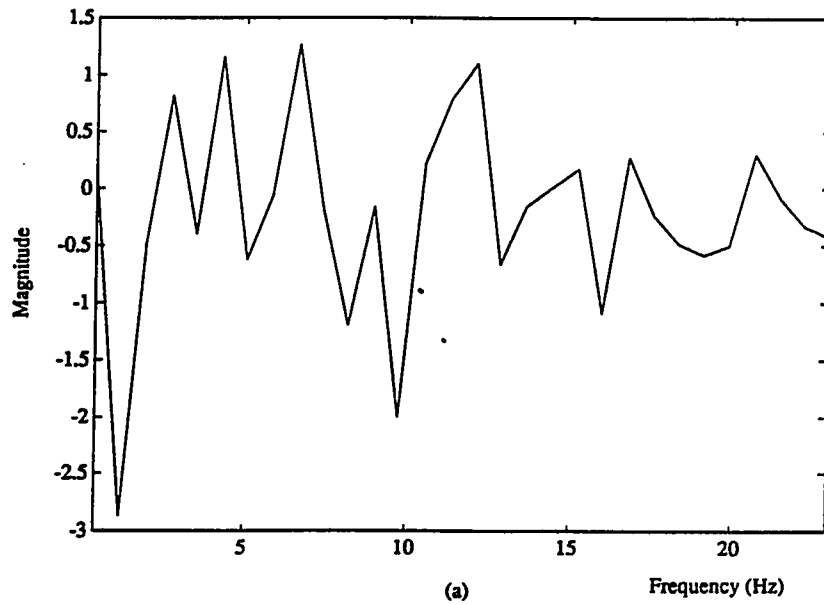


Figure 4.7: Weight Magnitude vs. Frequency for Middle Layer Neurode 1 for Networks Trained With Power and Phase Data Alone, and With Combined Data. Middle Layer Size Is Four Neurodes. (a) Power-trained Network. (b) Corresponding Inputs for Power Data for Combined Data-trained Network.

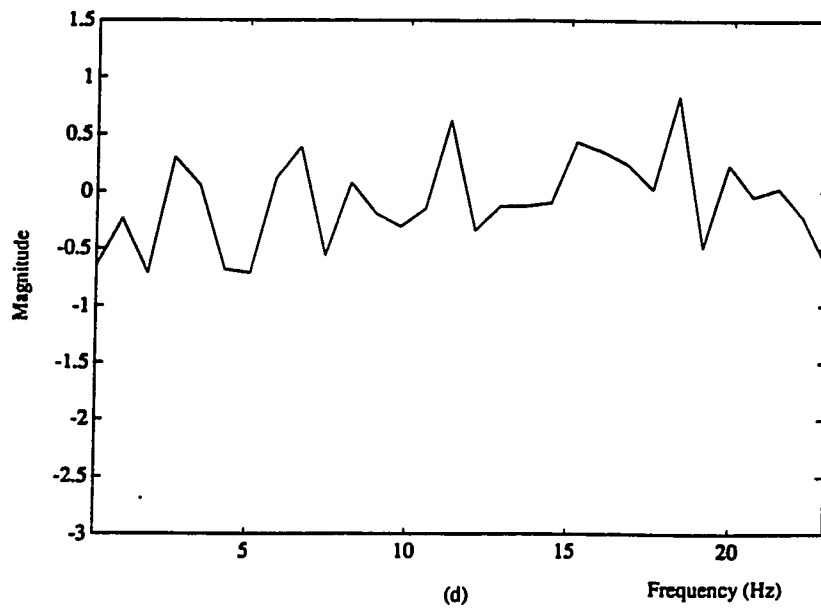
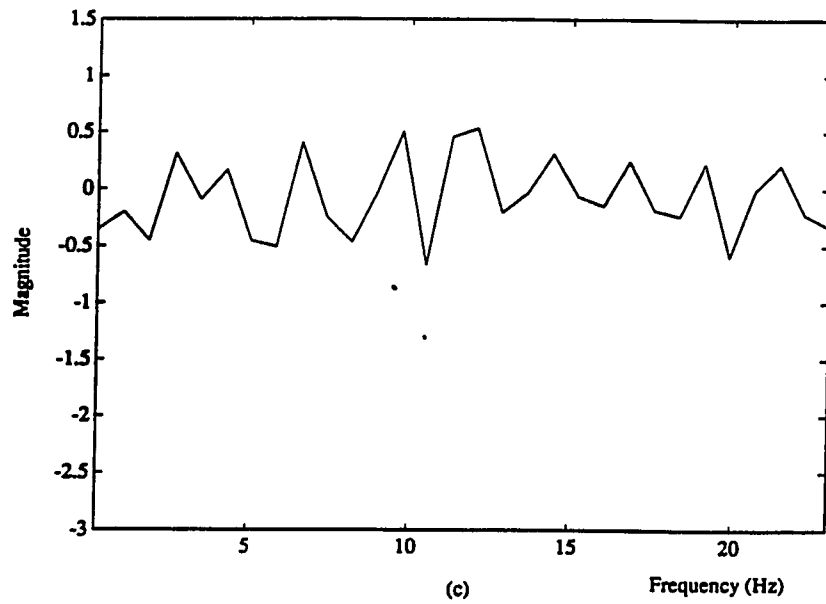


Figure 4.7 (continued) (c) Phase-trained Network. (d) Corresponding Inputs for Phase Data for Combined Data-trained Network.

In fact, the relationship could be shown as

$$I = \frac{a}{\eta^b} + c \quad (4.1)$$

where I is the number of iterations to convergence. This agrees with results reported by Higashino *et al* [16]. This relationship held until η reached a value where the weight update algorithm began to oscillate or the neurodes became saturated, and was particularly true for Phase and Combined data when $\theta_3 < 5.0$. Using regression analysis to solve for the constants in equation (4.1), b was found to be roughly “1” for all network characteristics, while a and c varied with middle layer size and group number. Figure 4.8 (a)–(l) demonstrates the network behaviour as η is increased for various values of θ_2 and θ_3 and middle layer sizes (“Average Iterations” is used in Figure 4.8 to denote that values given are averaged over the entire range of α in order to eliminate α as a variable). Note from Figure 4.8 that if $\theta_3 = 5.0$, iterations increase as η increases.

The examples shown in Figure 4.9 demonstrate the effect on Phase training of increasing η for a given network topology. Figure 4.9(a) shows a smooth, although relatively long, training period for $\eta = 0.1$. When η is increased to 0.3, training time drops considerably, as shown in (b). Further increasing η to 0.9 only decreases iterations slightly and creates oscillations. Finally, adding a small amount of momentum smooths the oscillations and decreases training time, as demonstrated in 4.9(d) (see section 4.1.1.7). Figure 4.10 shows that the feature extraction for these networks was the same for all the values of η and α , demonstrating that decreasing iterations by changing η and α has no effect on feature extraction.

Figure 4.11 demonstrates effects on Power training of increasing η for a given network topology. Note that training oscillates heavily in the initial iterations. Increasing η does not necessarily lead to faster training, as shown. Even adding momentum does not ensure decreased oscillations or training time. In fact, (d) shows that increasing momentum has actually led to saturation of the output neurodes (Figure 4.11(d) was the only network shown that did not converge in under 30,000 iterations). Figure 4.12 shows that the feature extraction was different

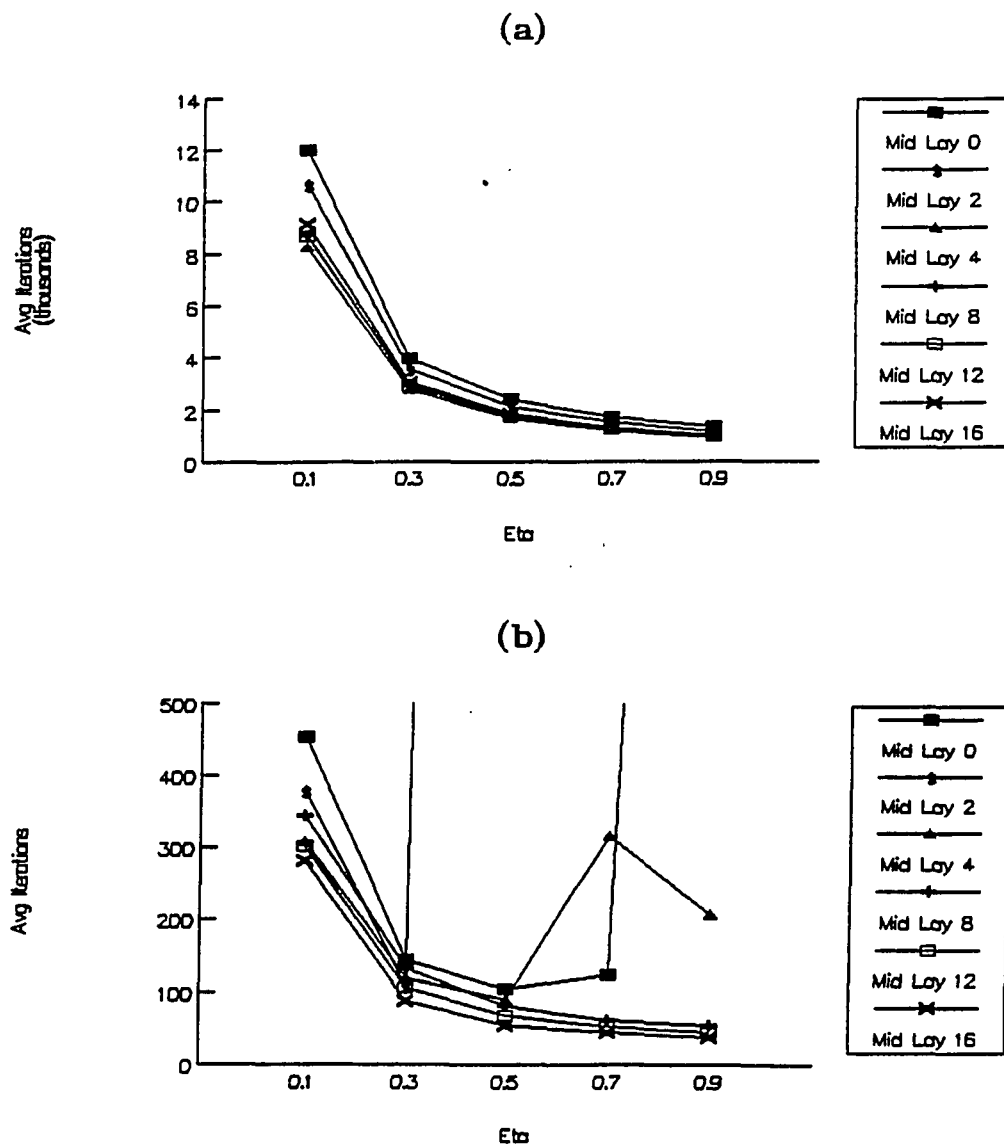


Figure 4.8: Average Iterations vs. η for Phase-trained Networks for Different Combinations of θ . (a) $\theta_2 = \theta_3 = 0.2$. (b) $\theta_2 = \theta_3 = 1.0$. (Group1)

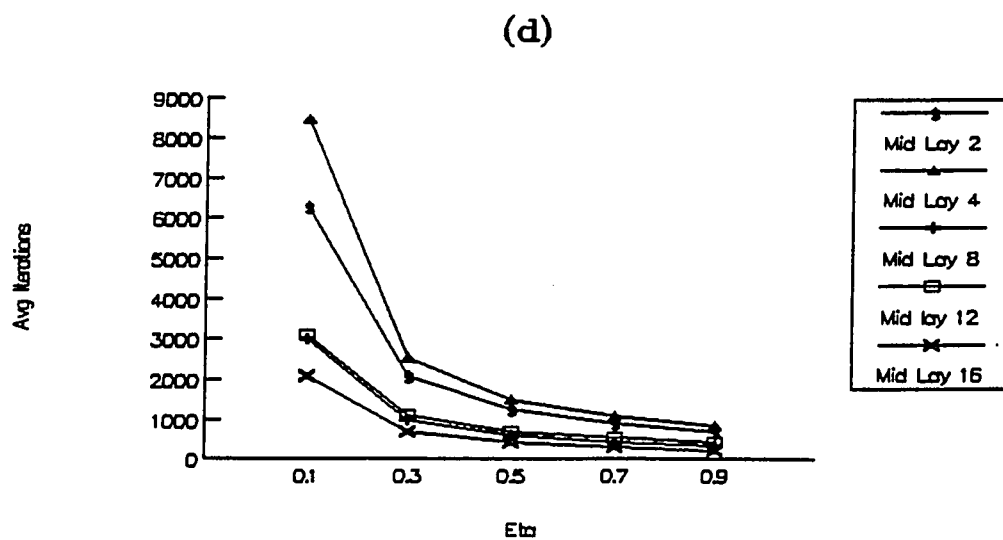
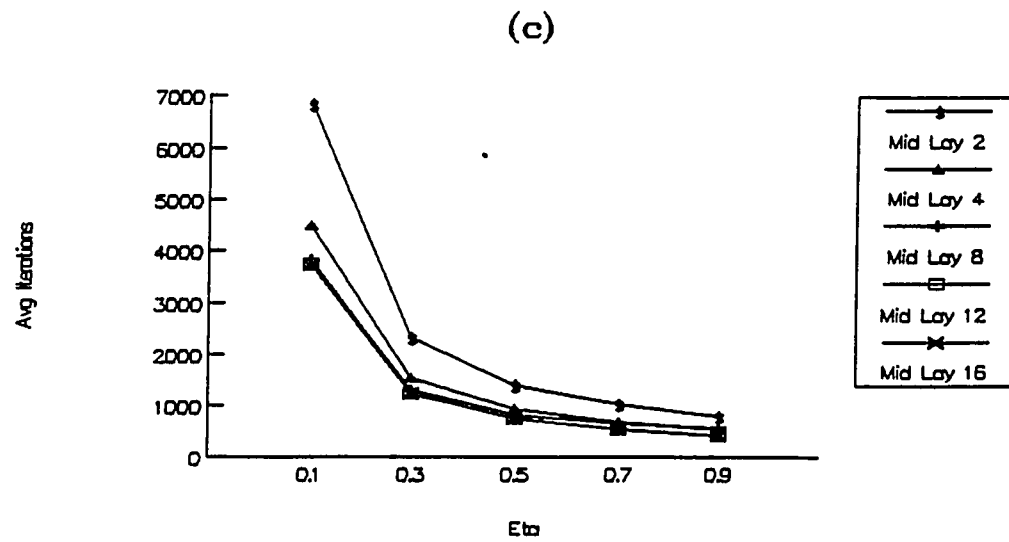


Figure 4.8 (continued). (c) $\theta_2 = 1.0$, $\theta_3 = 0.2$. (d) $\theta_2 = 5.0$, $\theta_3 = 0.2$.
(Group1)

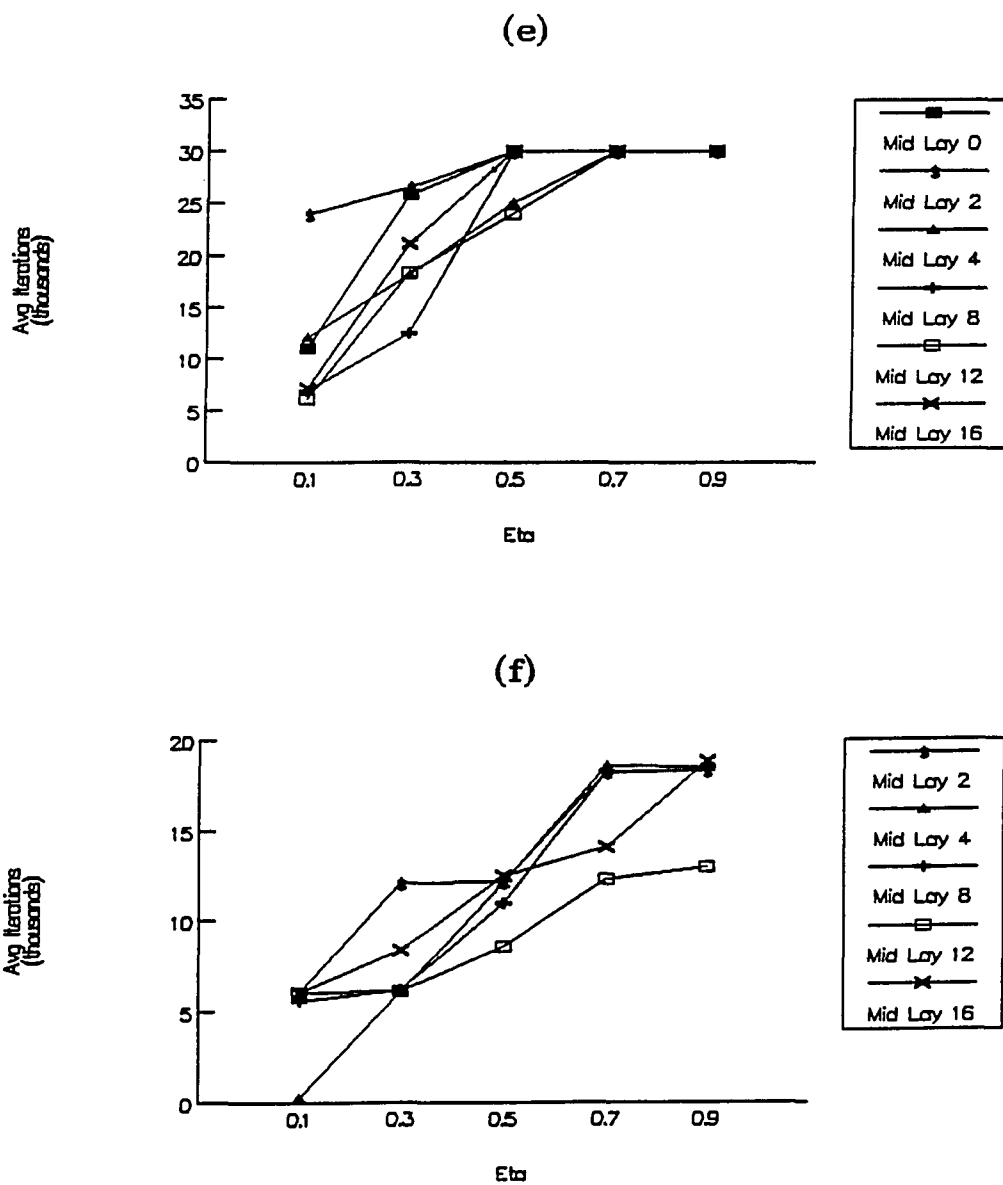


Figure 4.8 (continued). (e) $\theta_2 = 5.0$, $\theta_3 = 5.0$. (f) $\theta_2 = 0.2$, $\theta_3 = 5.0$.
(Group1)

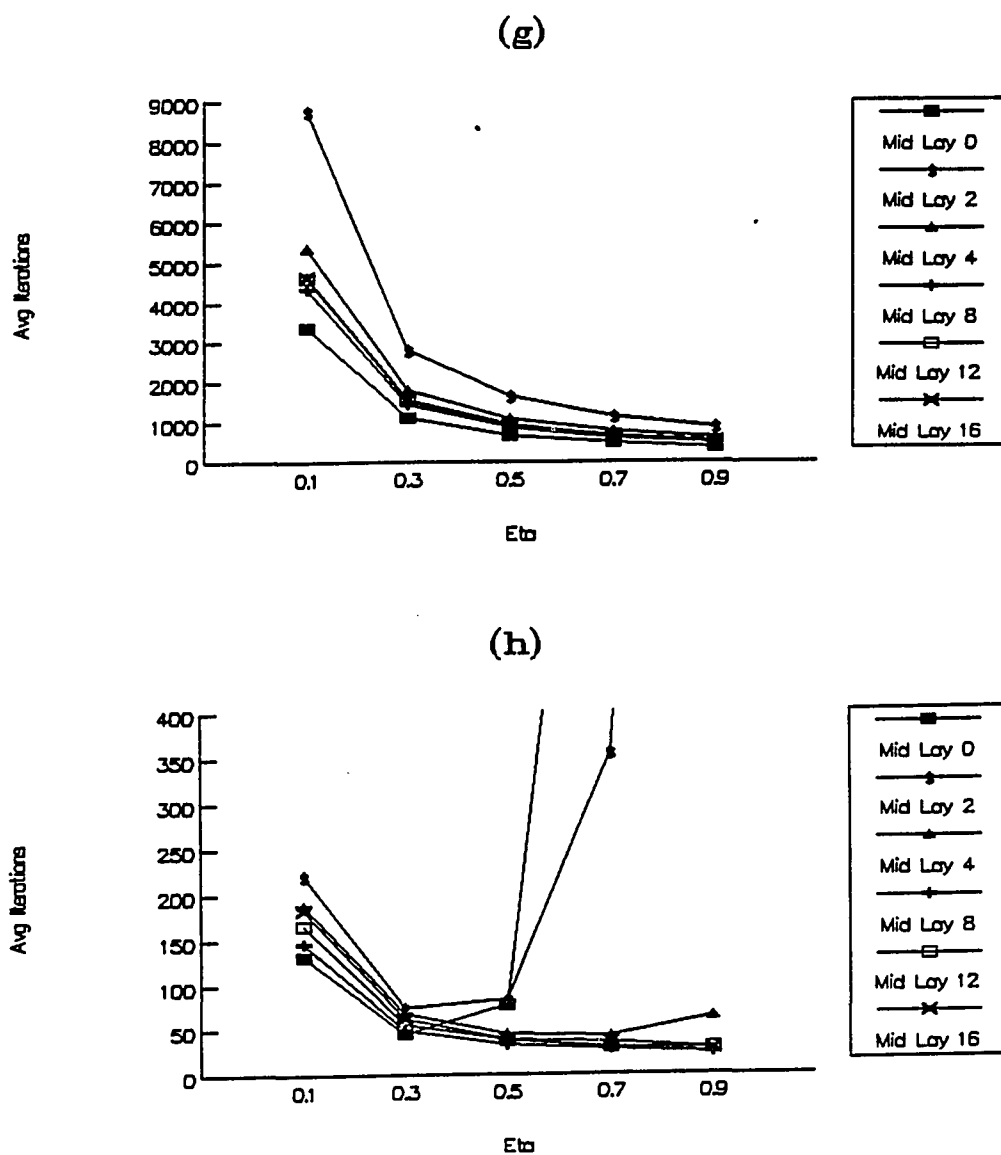


Figure 4.8 (continued). (g) $\theta_2 = \theta_3 = 0.2$. (h) $\theta_2 = \theta_3 = 1.0$. (Group2)

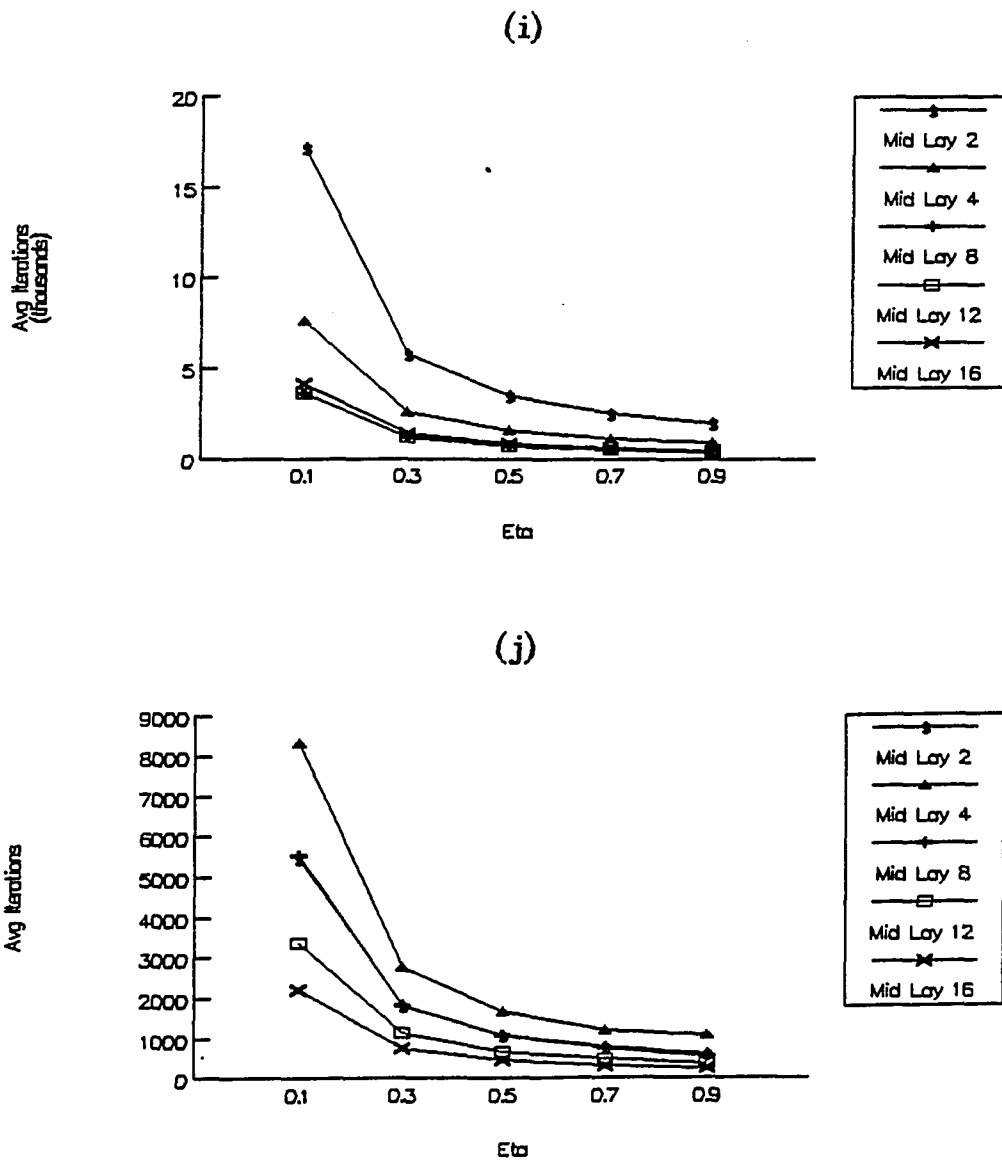


Figure 4.8 (continued). (i) $\theta_2 = 1.0, \theta_3 = 0.2$. (j) $\theta_2 = 5.0, \theta_3 = 0.2$.
(Group2)

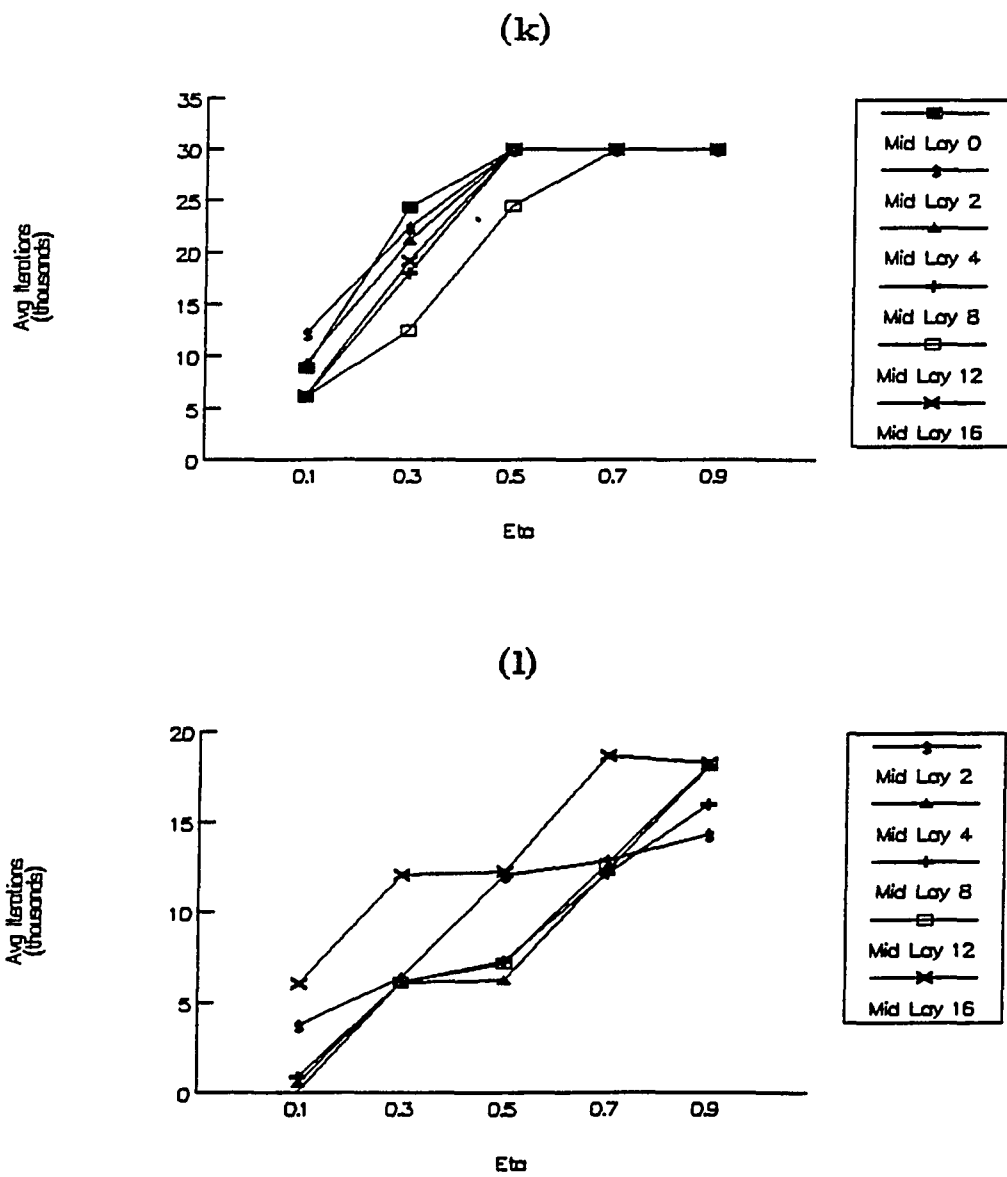


Figure 4.8 (continued). (k) $\theta_2 = 5.0, \theta_3 = 5.0$. (l) $\theta_2 = 0.2, \theta_3 = 5.0$.
 (Group2)

for each network, particularly at the lower frequencies, although some similarities did occur at certain frequencies.

4.1.1.7 Momentum Constant — α

The momentum constant also had a fairly consistent effect on the training performance of the networks. That is, as α increased, iterations to convergence decreased linearly. The relationship could be shown as

$$I \propto a - \alpha \quad (4.2)$$

which agrees again with Higashino *et al's* results [16]. The relationship held over nearly the entire range of α , and was particularly true for training with Phase and Combined data when $\theta_3 < 5.0$. The examples shown in Figure 4.13 (a)–(1) demonstrate the network behaviour as α is increased for various values of θ_2 and θ_3 and middle layer sizes ("Average Iterations" is used in Figure 4.13 to denote that values given are averaged over the entire range of η in order to eliminate η as a variable).. Note in Figure 4.13 that if $\theta_3 = 5.0$, iterations increase as α increases.

Referring back to Figure 4.9, note the smoothing effect of adding α during Phase-training. In contrast, Figure 4.11 shows that adding α may not consistently provide this smoothing or decrease training time, depending on the type of data used.

4.1.1.8 Activation Function Constant — θ

The activation function constant had a varied effect on training performance depending upon the data type. For Phase and Combined data, training performance was much more consistent for $\theta = 0.2$, while $\theta = 1.0$ allowed faster training. $\theta = 5.0$ tended to saturate the neurodes much more readily, preventing convergence. Networks with middle layer θ (θ_2) greater than output layer θ (θ_3) showed better consistency and convergence over the entire range of topologies with increasing η and α than those with opposite characteristics (see Figures 4.7 and 4.13). Figure 4.14 demonstrates average convergence times over the entire range of topologies

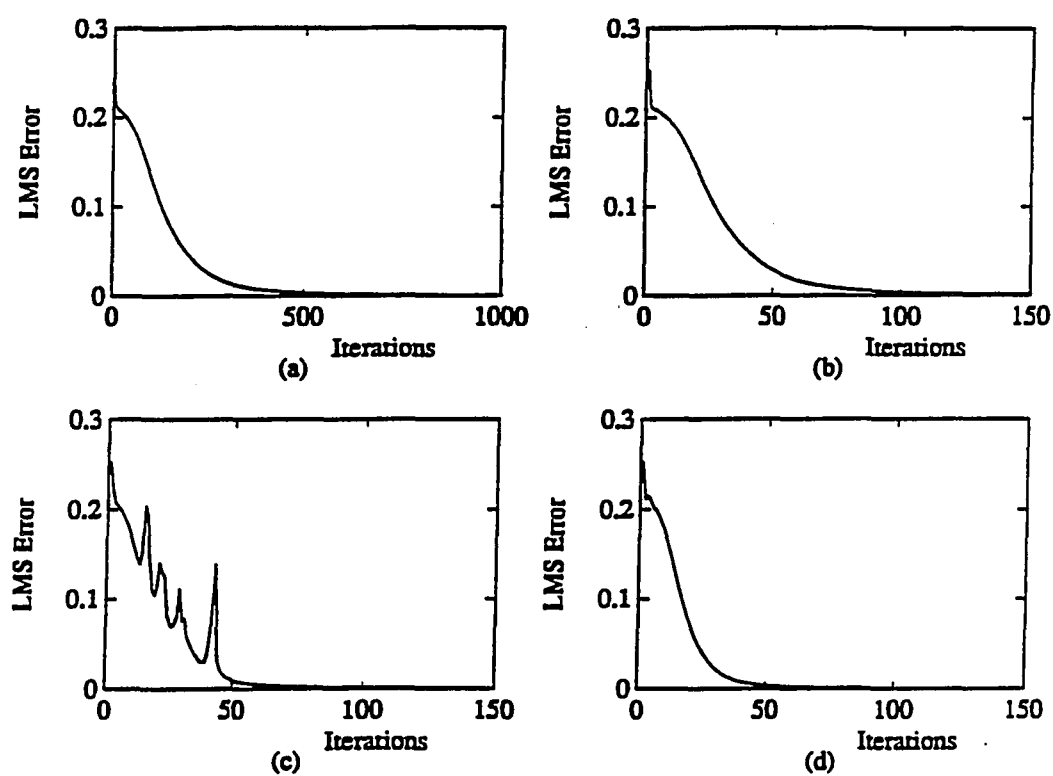


Figure 4.9: LMS Error vs. Iterations for Phase Data Training. (a) $\eta = 0.1$, $\alpha = 0.0$. (b) $\eta = 0.5$, $\alpha = 0.0$. (c) $\eta = 0.9$, $\alpha = 0.0$. (d) $\eta = 0.9$, $\alpha = 0.5$. For (a)–(d), $\theta_2 = \theta_3 = 1.0$ and middle layer size is 4. Note change in x-scale for (a).

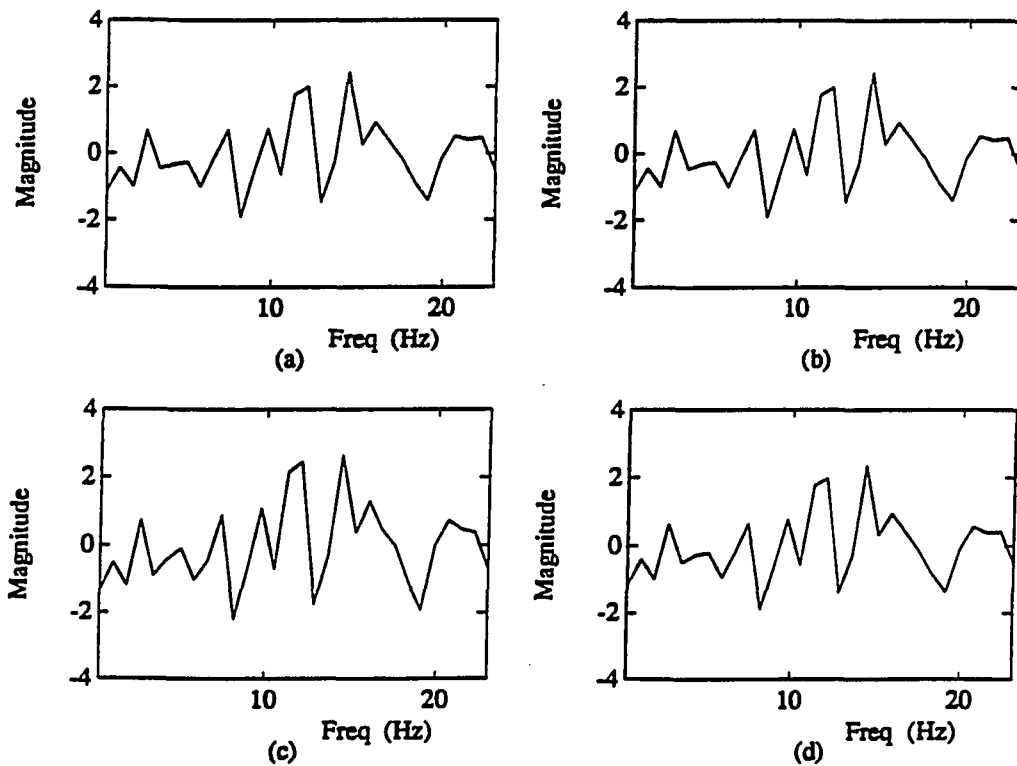


Figure 4.10: Weight Magnitude vs. Frequency for Middle Layer Neurode 1 for Phase Data Training. (a) $\eta = 0.1$, $\alpha = 0.0$. (b) $\eta = 0.5$, $\alpha = 0.0$. (c) $\eta = 0.9$, $\alpha = 0.0$. (d) $\eta = 0.9$, $\alpha = 0.5$. For (a)-(d), $\theta_2 = \theta_3 = 1.0$ and middle layer size is 4.

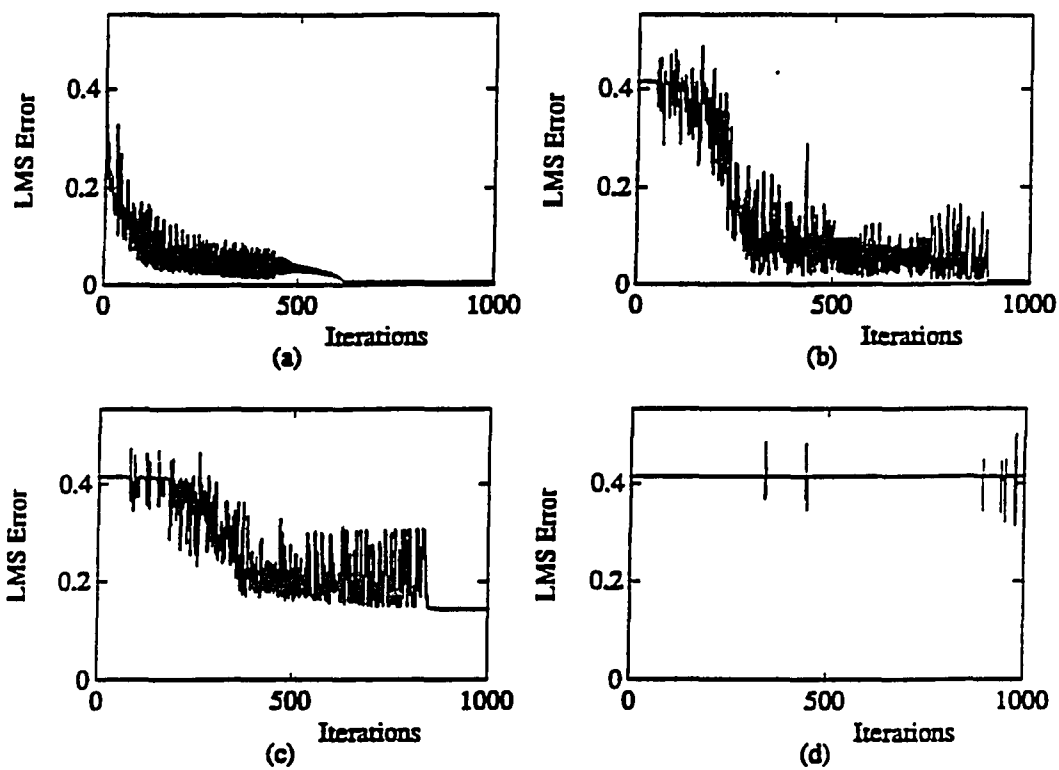


Figure 4.11: LMS Error vs. Iterations for Power Training. (a) $\eta = 0.1, \alpha = 0.0$. (b) $\eta = 0.3, \alpha = 0.0$. (c) $\eta = 0.3, \alpha = 0.1$. (d) $\eta = \alpha = 0.3$. For (a)–(d), $\theta_2 = \theta_3 = 5.0$ and middle layer size is 4. Note that only the first 1,000 iterations are shown.

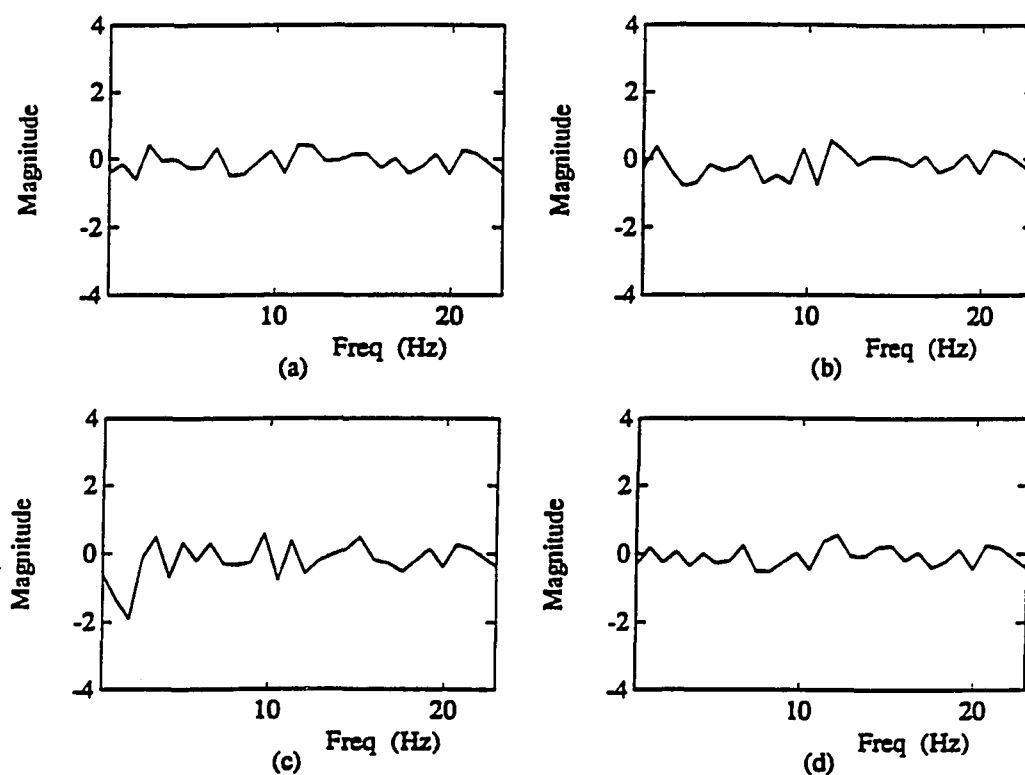


Figure 4.12: Weight Magnitude vs. Frequency for Middle Layer Neurode 1 for Power Training. (a) $\eta = 0.1$, $\alpha = 0.0$. (b) $\eta = 0.3$, $\alpha = 0.0$. (c) $\eta = 0.3$, $\alpha = 0.1$. (d) $\eta = \alpha = 0.3$. For (a)-(d), $\theta_2 = \theta_3 = 5.0$ and middle layer size is 4.

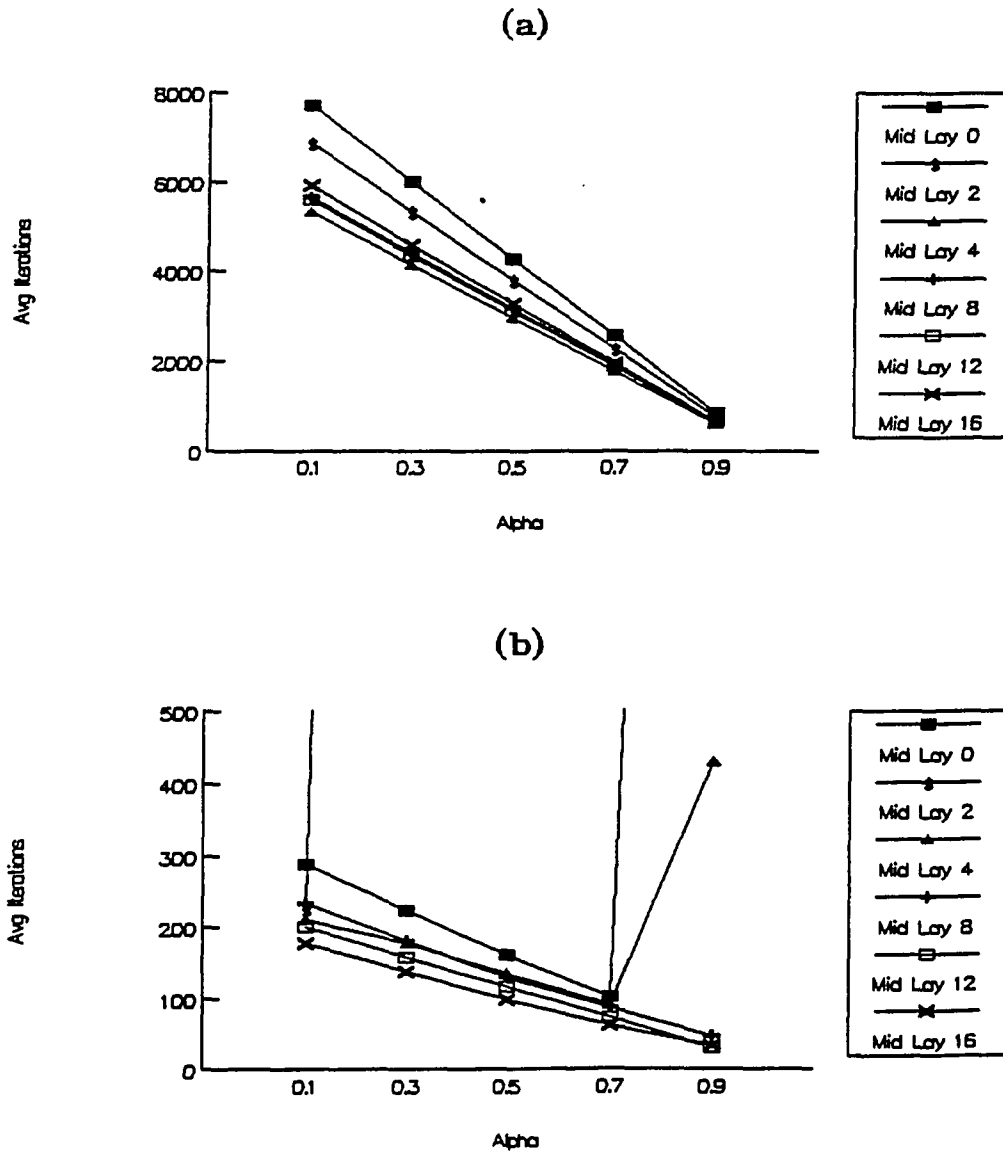


Figure 4.13: Average Iterations vs. α for Phase-trained Networks for Different Combinations of θ . (a) $\theta_2 = \theta_3 = 0.2$. (b) $\theta_2 = \theta_3 = 1.0$. (Group1)

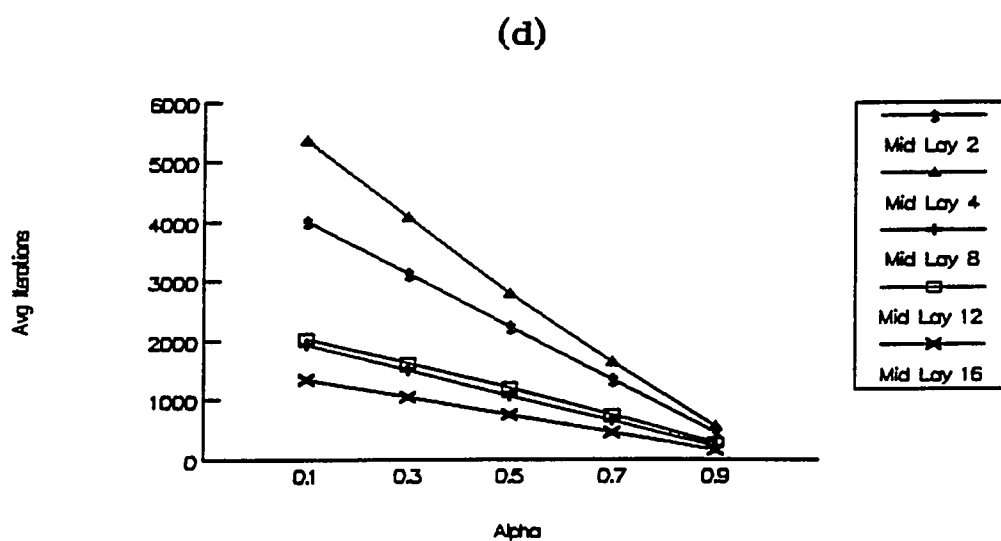
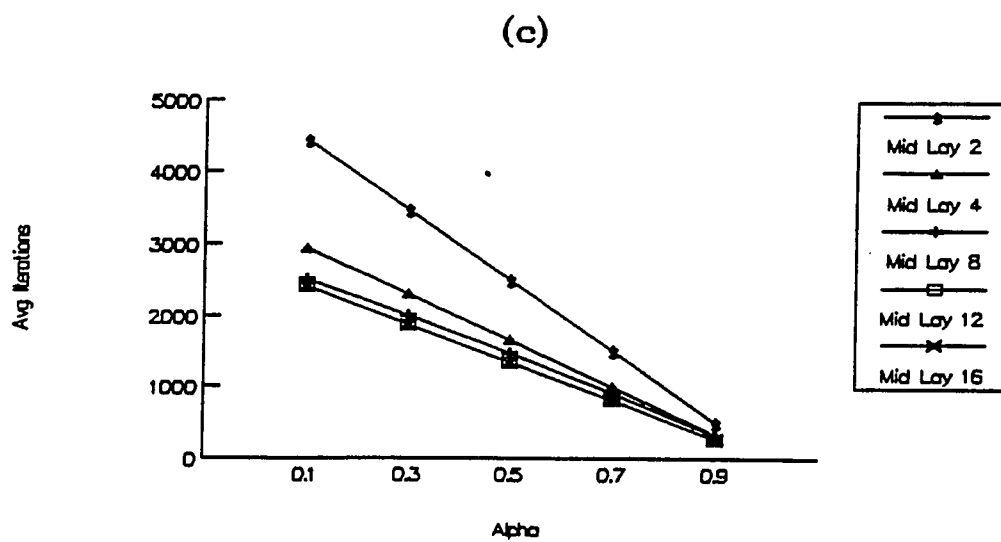


Figure 4.13 (continued). (c) $\theta_2 = 1.0$, $\theta_3 = 0.2$. (d) $\theta_2 = 5.0$, $\theta_3 = 0.2$.
(Group1)

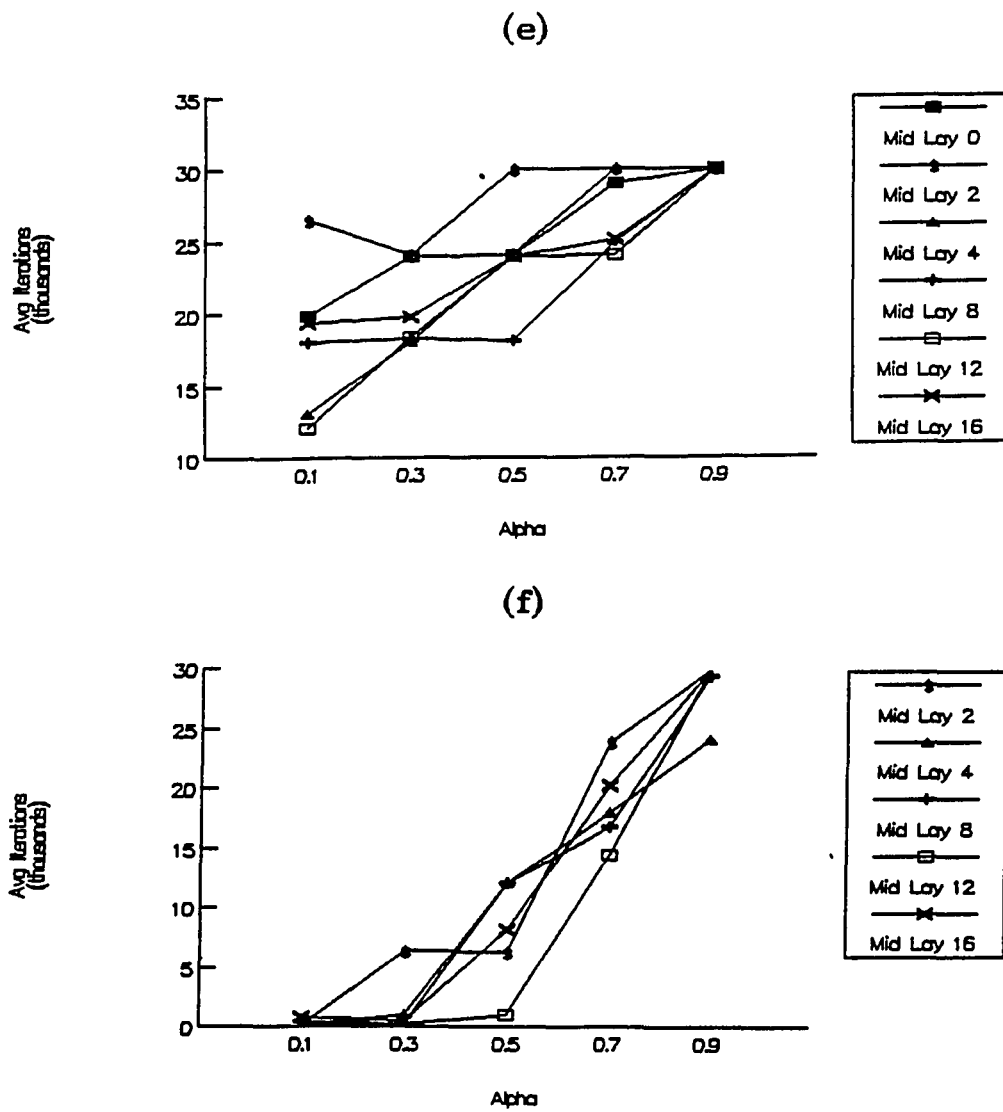


Figure 4.13 (continued). (e) $\theta_2 = 5.0$, $\theta_3 = 5.0$. (f) $\theta_2 = 0.2$, $\theta_3 = 5.0$.
(Group1)

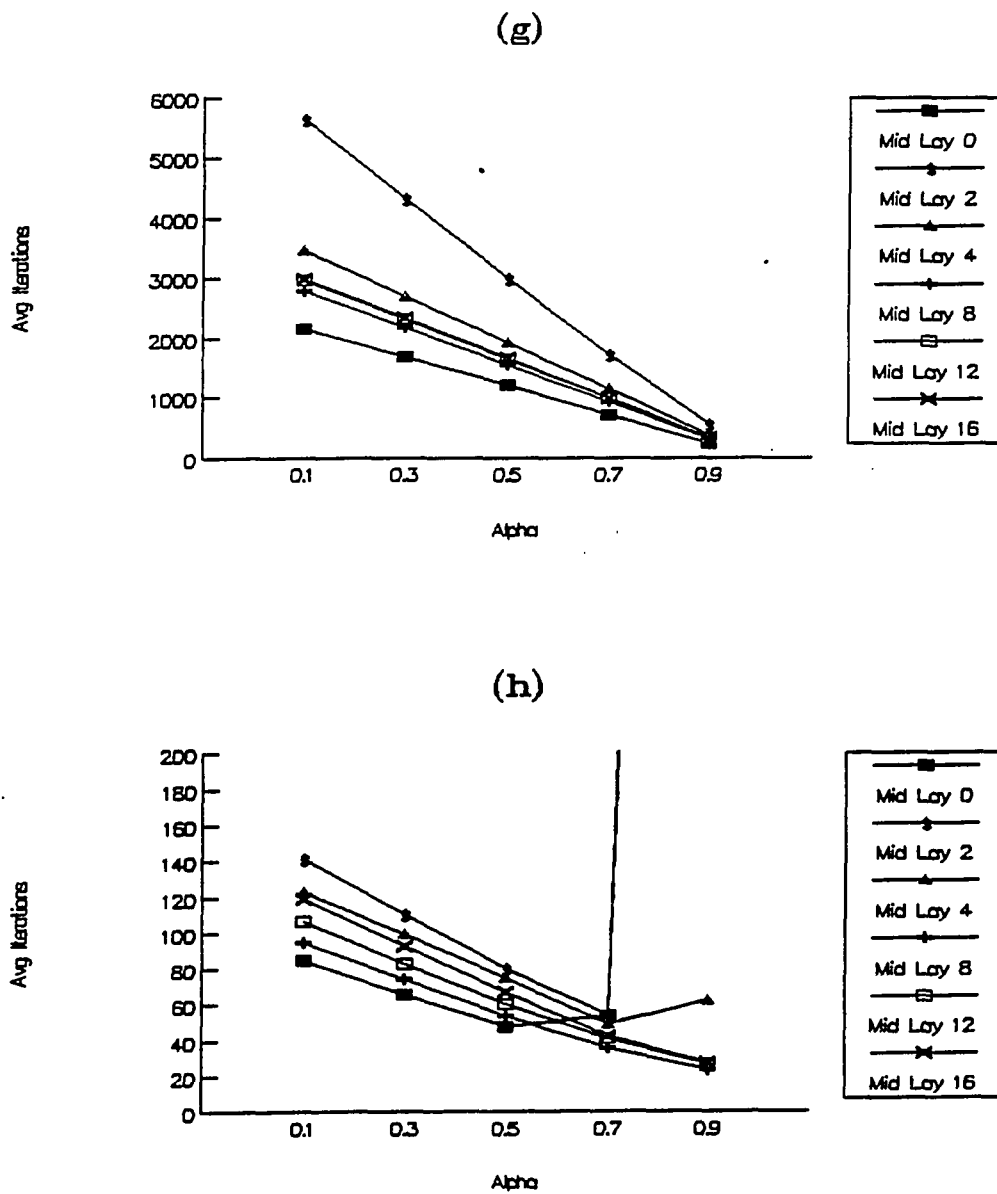


Figure 4.13 (continued). (g) $\theta_2 = \theta_3 = 0.2$. (h) $\theta_2 = \theta_3 = 1.0$. (Group2)

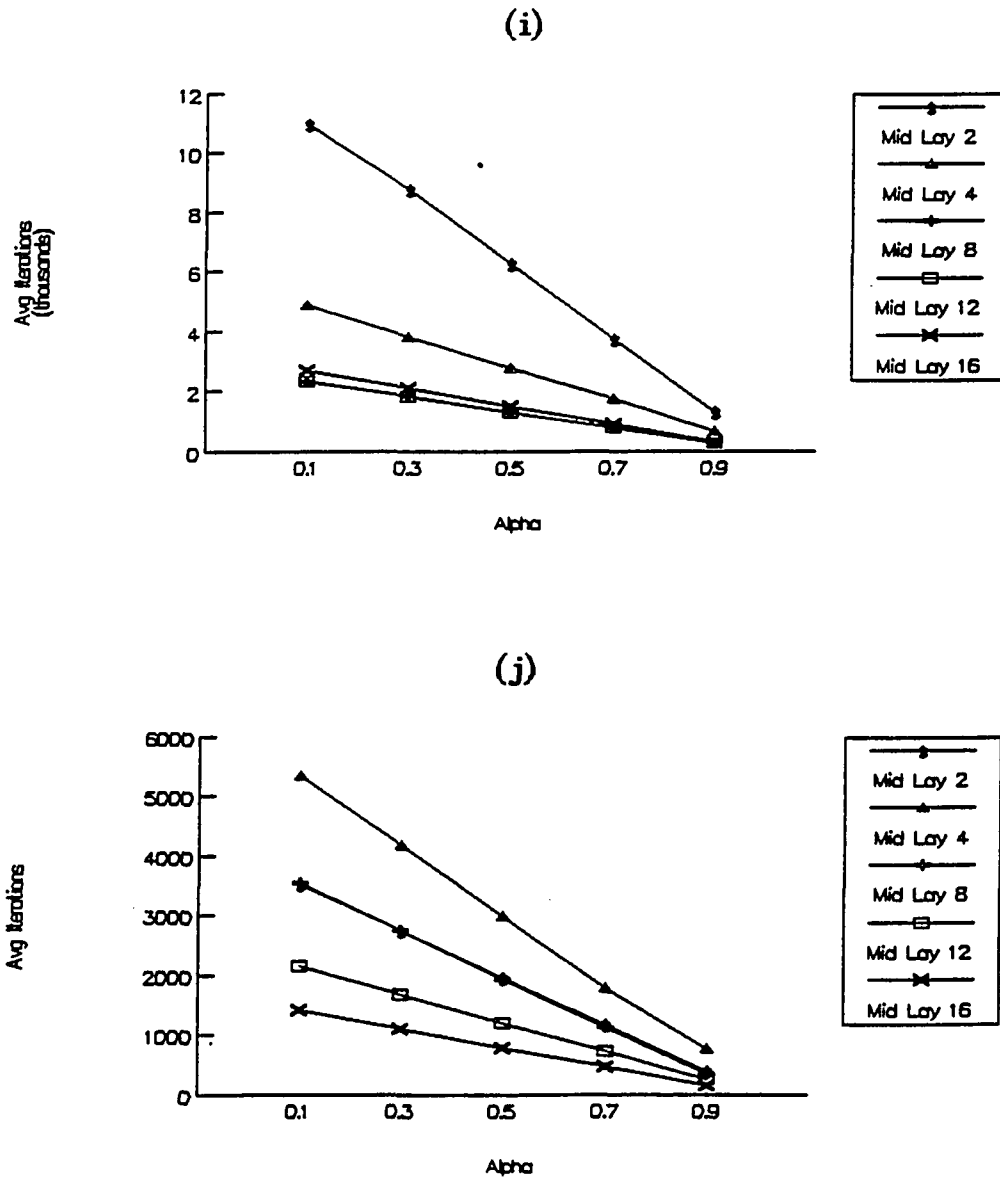


Figure 4.13 (continued). (i) $\theta_2 = 1.0$, $\theta_3 = 0.2$. (j) $\theta_2 = 5.0$, $\theta_3 = 0.2$.
(Group2)

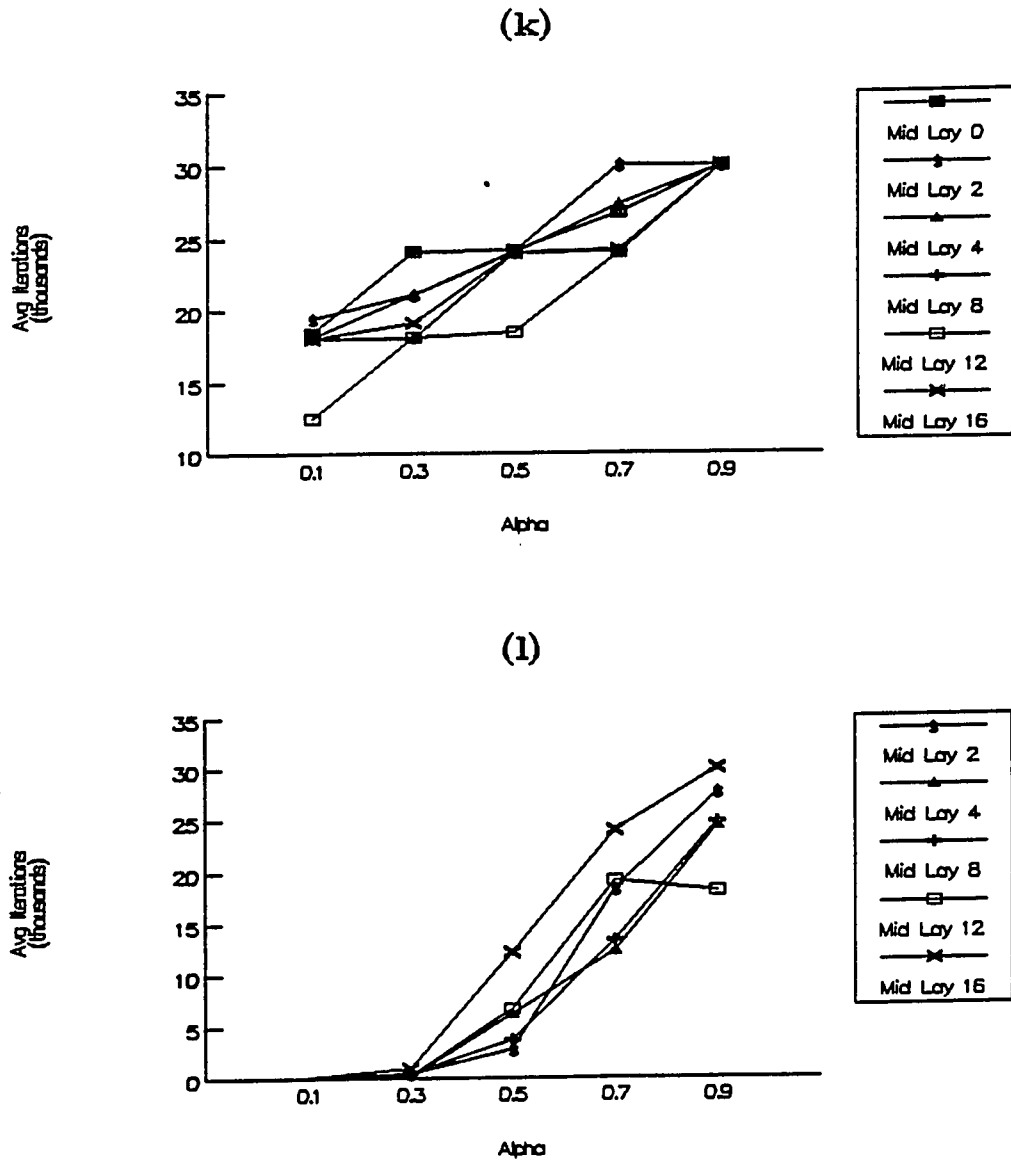


Figure 4.13 (continued). (k) $\theta_2 = 5.0, \theta_3 = 5.0$. (l) $\theta_2 = 0.2, \theta_3 = 5.0$.
 (Group2)

for various combinations of θ_2 and θ_3 . "Average Iterations" is used to denote that values given are over the entire range of middle layer sizes in order to eliminate that as a variable. Note that iterations for $\theta_2 = \theta_3 = 1.0$ increases as η and α increase, while the opposite is true for other combinations of θ .

For Power data, it was nearly impossible to converge when $\theta_2 = \theta_3 = 0.2$ if middle layer size was less than 4 neurodes or η and α were less than 0.5. When $\theta_2 = \theta_3$, the best performance was with $\theta = 5.0$. The most consistent and convergent performance over the entire range of topologies with increasing η and α was when $\theta_2 = 5.0$ and $\theta_3 = 0.2$, as shown in Figure 4.15. Notice that for these values of θ , iterations are almost completely independent of middle layer size, showing a similar response to Phase-trained networks.

4.1.1.9 Training Performance Criteria

Maximum iterations, T , was always set to 30,000. For the Phase and Combined data, this turned out to be a good number: virtually all the networks could train in this amount of time. For the Power data, approximately 40% of the networks were able to train in the allotted time.

Increasing the error criteria, ϵ , always decreased the number of iterations to convergence, as expected. The more important effect of increasing ϵ will be discussed in classification section 4.2.1.7.

4.1.2 Discussion (implications)

The most significant fact observed in the BPN training research is that different data types provide different performance results over variations in network characteristics. Phase data, which has relatively small fluctuations in peak magnitudes, has much more stable performance. Because of the small fluctuations in peak magnitudes, Phase data responds much better to smaller values of θ . Setting $\theta = 0.2$ allows the neurodes to remain in the dynamic portion of the Activation Function during the *forward* pass of the Generalized Delta Rule (accounting for the more stable performance for $\theta = 0.2$), while setting $\theta = 1.0$ allows faster training during the backpropagation pass of the GDR. Using large values of η decreases

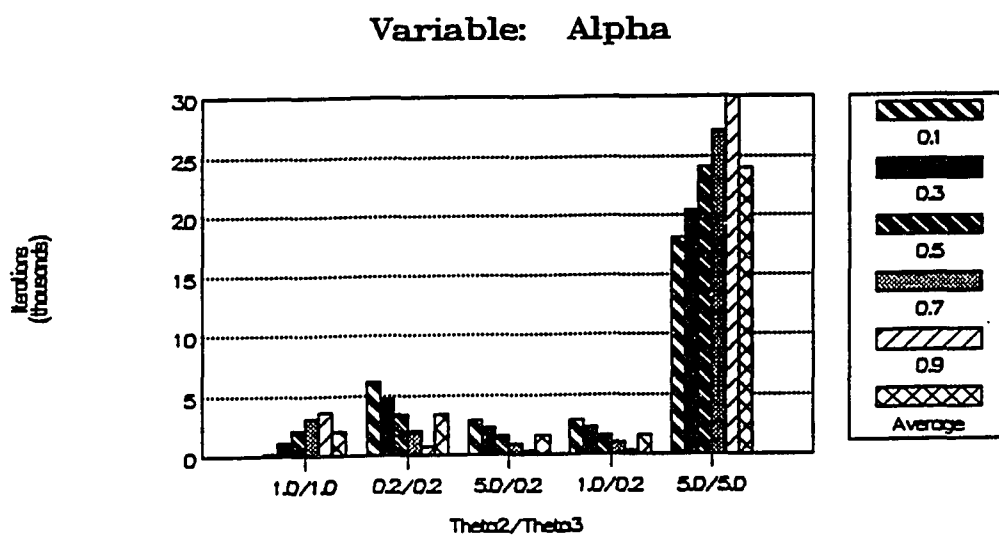
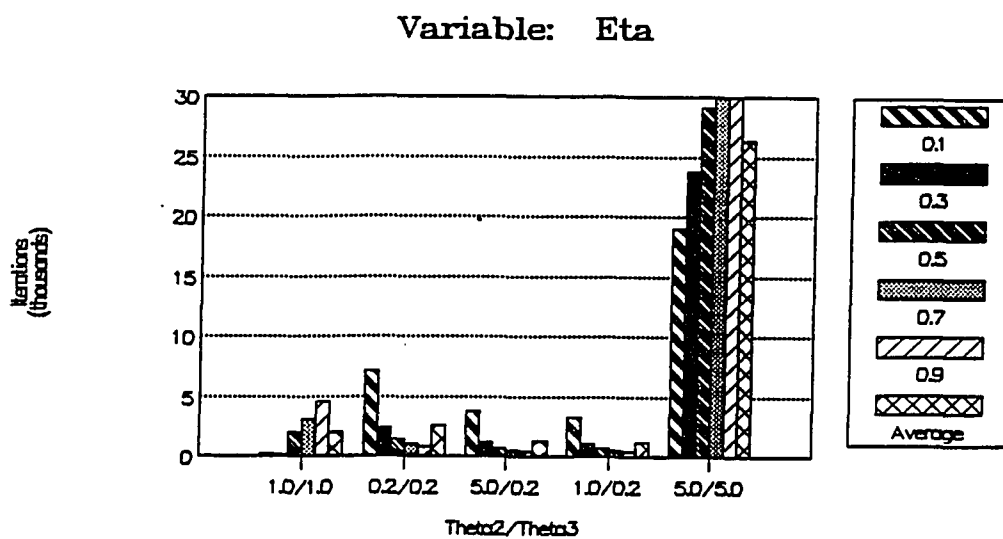


Figure 4.14: Iterations vs. Various Combinations of θ for Phase-Trained Networks. (a) η is variable (note legend to right of graph). (b) α is variable (note legend).

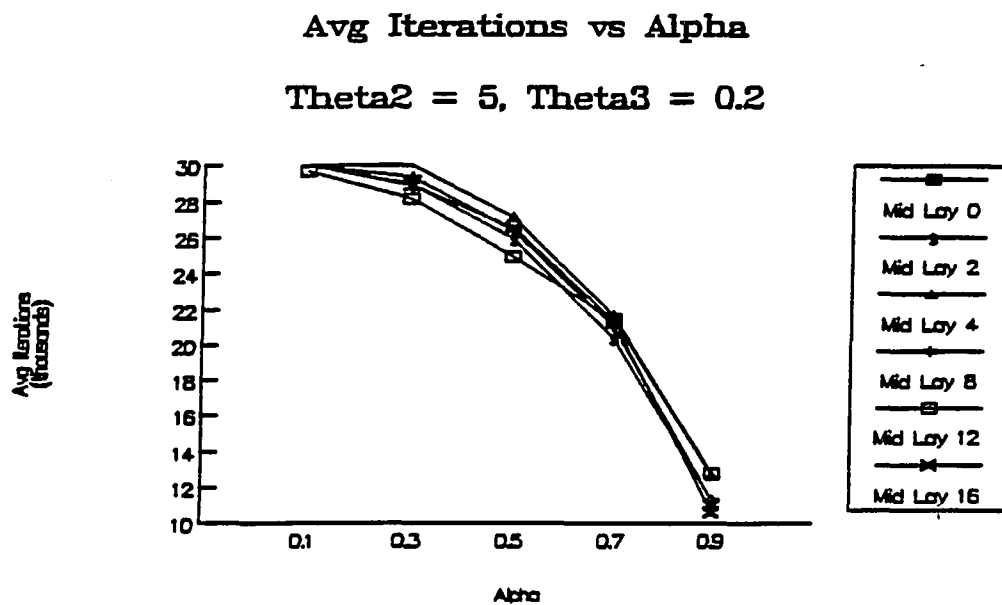
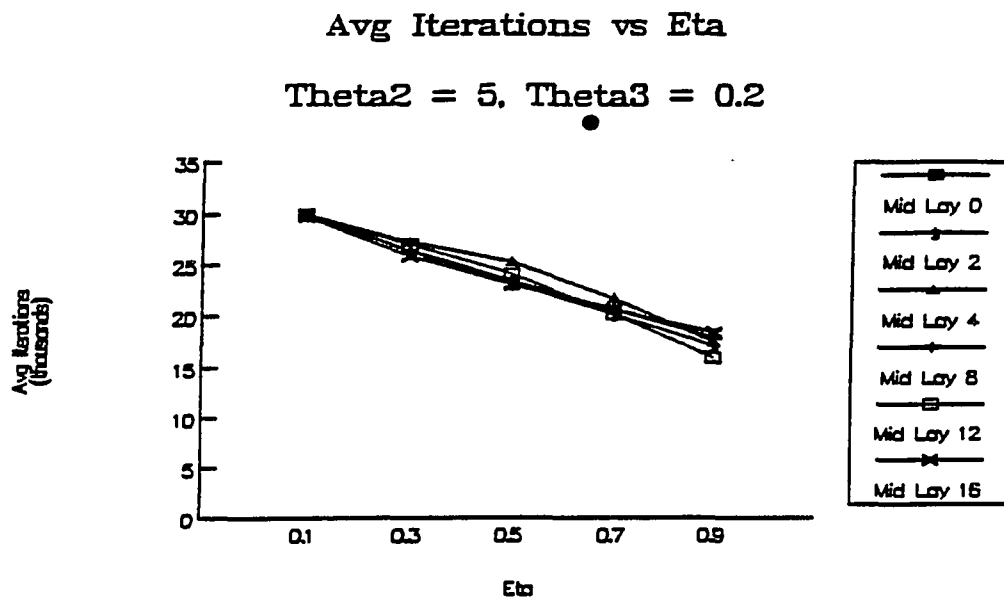


Figure 4.15: Average Iterations vs. η and α for $\theta_2 = 5.0$ and $\theta_3 = 0.2$ for Power Data. (a) Average Iterations vs. η . (b) Average Iterations vs. α . (Group1.)

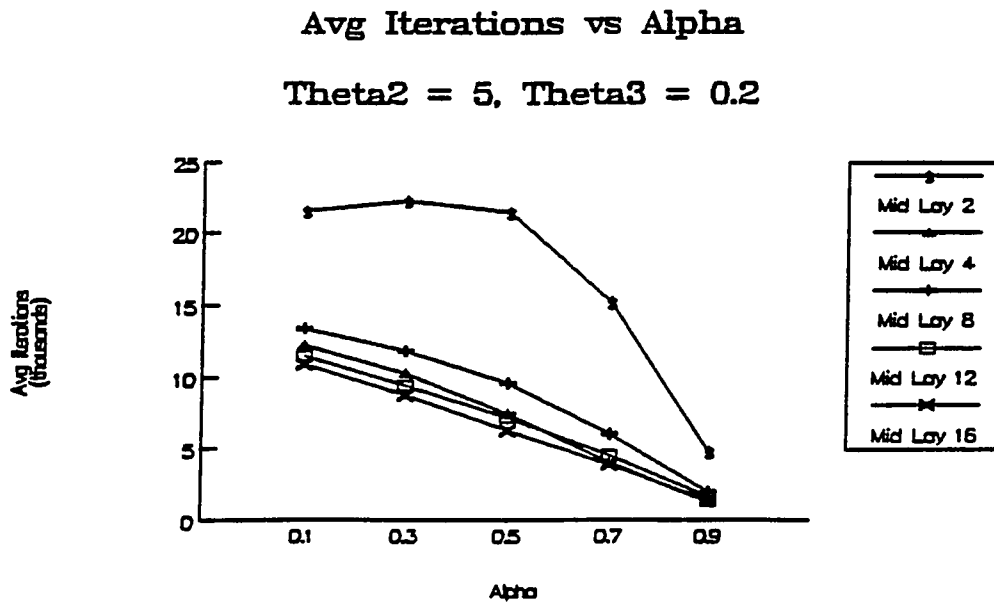
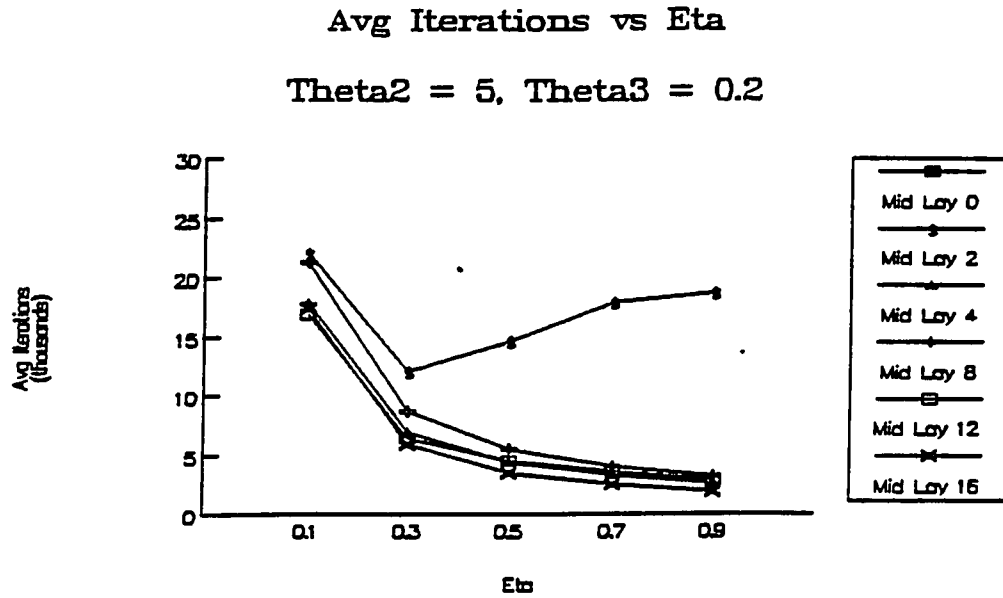


Figure 4.15 (cont.) (c) Average Iterations vs. η . (d) Average Iterations vs. α . (Group2.)

iterations for Phase training. This indicates, according to Caudill [5], that the patterns within the chosen groups are very similar, which further indicates that Phase data could be very useful in EEG monitoring.

Power data, which has relatively large fluctuations in peak magnitudes, has much less stable performance and is susceptible to neurode saturation. A lower value of θ appears to degrade the learning rate considerably for feature extraction in the first middle layer. This is particularly evident since $\theta_2 = 5.0$ and $\theta_3 = 0.2$ provides the best training for Power data. Large values of η tend to send the networks into saturation, indicating that the patterns within the chosen groups are very dissimilar [5].

Combining the data and presenting it together to a network provides no training advantage over Phase alone, but shows a significant increase in performance over Power alone. Performance characteristics seem to mimic those for Phase alone. It was shown that the networks are almost completely rejecting the Power data and only using the Phase data to update neurode weights. This would indicate that this method of combining different data types for presentation to network is not particularly useful.

The differences in training performance between groups are readily apparent. The increased training time for Group1 over Group2 indicates that the subjects chosen for Group1 have less homogeneous features for a given pattern class. The variations in Phase data, however, are not so severe that a network could not train over the entire set of subjects: a combined group of Group1 and Group2 subjects was able to train a single middle layer BPN in an average of 450 iterations, with a low of 107 iterations. The same could not be said for the Power data.

The type of data used is the most important factor in determining whether adding a second middle layer decreases iterations to convergence. The Power data benefits greatly from a second middle layer, while iterations increase and stability decreases for Phase data. This would indicate that the different classes of Power data (MAC 0.5–1.5) have non-linearly separable patterns, while the Phase data do not.

The type of data used is, again, the most important factor in determining whether increasing the middle layer size(s) decreases iterations to convergence. The Power data benefits by increasing the number of neurodes up to some point, while Phase data shows much less benefit. For BPNs with two middle layers, the importance of the relative sizes of the middle layers is also dependent on the data type. Since Power data performs better with larger (first) middle layers, this would indicate that this data type has less varied features which requires more feature extraction neurodes in order to discover the uniqueness between classes. Increasing middle layer size, however, works in opposition to the desire for a network that will generalize for patterns it has yet to see. This will be discussed in the Classification section below.

The multiple network BPN shows good performance characteristics over a wide range of network parameters. I believe it is more appropriate to train separate networks with different types of data, rather than one network with different types of data combined. The output from these separate networks can be made identical in character, as was done here. The final network is then a very simple BPN that can have zero or one middle layer(s). The importance of the multiple network BPN will depend on classification results, which will be discussed in the Classification section below.

It is satisfying to see that the learning constant, η , has a fairly consistent effect on the training performance. The model of this performance, $I \propto 1/\eta$, agrees with results from Higashino *et al* [16], which reinforces the validity of the methods and results. It is interesting to note that, as expected, increasing η will at some value create oscillations in training, and may even cause saturation. The I - η relationship will be useful in the future for finding values of η where further increasing η causes no benefit. The relationship cannot, however, tell where training will become unstable.

Like η , it is satisfying to see that the momentum constant, α , has a fairly consistent effect on the training performance. The model of this performance, $I \propto a - \alpha$, also agrees with results from Higashino *et al* [16], further reinforcing the research validity. The I - α relationship demonstrates the advantages of increasing

α up to the point where saturation occurs; unfortunately, the relationship cannot predict where this point is. The enhanced performance provided by the momentum fairly negates Pao's comments that it slows the training rate [25] and Von Lehmen *et al's* view that the extra memory required by adding momentum outweighs its usefulness [43].

Contrary to Caudill's assertion [5] that network performance is independent of the activation function constant, θ , this research found that the value is extremely important to training results. While the steeper AF slope may provide faster training, it also leads to neurode saturation, as confirmed by Rezgui [34]. It was expected that a larger value of θ would provide better feature extraction because of the higher variability in the input data (range $[0 \dots \infty]$), while a smaller value would provide better classification in the output layer because of lesser variability in that layers input data (range $[0 \dots 1]$). This was proven by the result that, for all data types, $\theta_2 = 5.0$ and $\theta_3 = 0.2$ provided the most stable performance over variations in other network parameters. It is impossible to prove/disprove Izui and Pentland's theory [19] that iteration time was proportional to $1/\sqrt{\theta}$: Quite often the relationship is just the opposite.

4.2 BPN Classification Simulations

Roughly 500 of the networks formed for training were used for classification. These networks were selected for convergence properties and to obtain a large data base of similar network characteristics in order to compare and contrast testing capabilities. Classification results were obtained by training a BPN with one group of data and then measuring classification capabilities with the other group (see section 3.3). Results were categorized under "classification score" and "misclassification score". Classification score refers to the percentage of patterns correctly classified by the BPN. Misclassification score refers to percentage of patterns incorrectly classified. Since BPN output patterns were discrete values (i.e. [001], [010], [100]), outputs of the BPN could be (1) classified correctly if the output corresponded with *any* target value, and the output was the *expected* value ([001] \rightarrow [001]), (2) misclassified if the output corresponded with *any* target value, but the output was

not the expected value ([001] \rightarrow [010]), or (3) unclassified if the output *did not* correspond with any target value ([001] \rightarrow [011]). Therefore, a classification score of 60% for a BPN did not necessarily mean the misclassification score was 40%; in fact, as will be shown, patterns that were not correctly classified by the BPN typically were unclassified.

4.2.1 *Results*

4.2.1.1 *Data Type — Power, Phase, Combined*

Although Phase and Combined data trained the networks faster with fewer cases of neurode saturation, BPNs trained with Power data had higher classification scores. Power data had average classification scores of 53% and highs of 73%. Phase had average scores of 47% and highs of 60%. Combined data averaged 40% with highs of 60%. Power data was much more consistent than Phase and Combined data in classification capabilities over a range of network characteristics (when networks converged).

4.2.1.2 *Group1 vs. Group2*

For Power data, networks trained with Group2 data had higher average classification scores than vice versa. The highest classification score was also obtained for a Group2-trained network classifying Group1. It is interesting to note that Group2 data trained the networks faster than Group1. It might be expected that since Group1 appeared to contain more dissimilar patterns, requiring more time to train, then a network trained on this data would be better able to classify a group of new patterns that appeared to be more homogeneous. This was obviously not the case.

For Phase data, networks trained with Group1 data had higher average classification scores, although a Group2-trained network displayed the single highest score. It is interesting to note here that Group2 data had better average training scores, but classification scores were worse — just the opposite of Power data. Thus, Phase data affirms the expectation that a network trained to generalize

patterns because of a relatively non-homogeneous training group (such as Group 1 data) should be able to better classify a more homogeneous group of new patterns (such as Group 2 data).

4.2.1.3 *Middle Layers*

Two middle layers provided much more consistency in classification scores than one middle layer. While doing little to increase *average* classification score for Phase data, having two middle layers did increase average scores for Power data. In fact, the highest score for Power data was achieved with two middle layers.

4.2.1.4 *Middle Layer Size*

The effect of middle layer size(s) was somewhat inconsistent between networks. For networks with one middle layer, changing the middle layer size had little, if any, consequence on the classification scores for Phase data. For the most part, classification scores dropped for Power and Combined data as middle layer size increased. For networks with two middle layers, classification scores were fairly independent of relative layer sizes. A large first middle layer followed by a small second middle layer worked equally as well as one with opposite characteristics.

4.2.1.5 *Multiple Network BPN*

The multiple network BPN provided the best classification capability for Combined data. Scores were very consistent for a variety of network characteristics. The highest Combined data score of 60% was achieved using this method.

4.2.1.6 *Activation Function Constant — θ*

Networks with larger θ values had better classification scores than those with lower values. Scores were also better when middle layer θ , θ_2 , was greater than output layer θ , θ_3 .

For a network trained with θ_T , increasing the constant during classification either increased the classification score, usually by 7% (75% of the cases), or left the score unchanged (25% of the cases). Decreasing θ had the opposite effect.

Increasing θ_3 had more advantageous results than increasing θ_2 , although increasing either by itself improved classification scores.

It was believed that increasing θ during training might lead to higher misclassification of patterns. This was not the case. In fact, misclassification remained at zero for virtually *all* the testing performed in this research. This was an extremely important result that will be discussed further in section 4.3 when comparing statistical analysis with BPN analysis.

4.2.1.7 Training Performance Criteria

The majority of training was performed with $\epsilon = 0.001$. Occasionally this value was increased to values given in section 3.2.5. Increasing ϵ by a factor of 5 to 0.005 increased classification scores by 7% for most tests. Any further increase in ϵ decreased classification scores, with $\epsilon = 0.625$ providing zero classification capability ($E = 1.215$ is the highest possible LMS Error, based on equation (2.4) and an output layer size of 3 neurodes). Increasing ϵ also had no effect on misclassification.

Convergence time did not appear to have any effect on classification score. That is, networks that took relatively long periods of time to converge did not classify patterns any worse/better than those BPNs that took relatively short periods of time.

4.2.1.8 Classification Performance Criteria

Data were classified according to two criteria:

Criteria 1 :

$$o_{pj} < .33 \Rightarrow o_{pj} = .1$$

$$o_{pj} \geq .67 \Rightarrow o_{pj} = .9$$

$$.33 \leq o_{pj} < .67 \Rightarrow o_{pj} = .5$$

Criteria 2 :

$$o_{pj} < .5 \Rightarrow o_{pj} = .1$$

$$o_{pj} \geq .5 \Rightarrow o_{pj} = .9$$

Classification scores using Criteria 2 were always better by roughly 7% than those for Criteria 1. Misclassification was identical, i.e. zero, for both sets of criteria.

4.2.1.9 *Learning and Momentum Constants*

Changing BPNs by using different η and α values had no noticeable consequence on classification scores.

4.2.2 *Discussion (Implications)*

Once again, the most important factor to BPN classification results is the data type, although unlike training, the best data type for *classification* is Power data. This is significant: even though the Power data takes considerably longer to converge on a network solution (or maybe *because* it takes longer), the networks formed are better able to classify new patterns. The networks formed using Power data are better able to generalize their decision boundaries, thereby recognizing new input patterns with different features from the training set. Therefore, because a certain data type provides better convergence properties than another does not necessarily mean it will also provide better classification capabilities. Power data is also more consistent in classification performance over a range of network characteristics.

Networks trained on Group2 Power data have better classification scores than those trained with Group1 data. This difference is normally 7%. This is just the opposite from Phase-trained networks, where BPNs trained on Group1 outperform those trained on Group2 during classification. Thus, it is impossible from this research to make a firm assertion as to the classification capabilities of BPNs whose training data is more or less homogeneous than others. A significant reason for the differences in the group performances may be that the size of the training set is relatively small compared with the classification set. None of the research specified in the literature used training sets that were the same size as (or smaller than) the classification sets. In fact, several of the researchers used classification sets that contained patterns from the training sets, obviously boosting performance scores.

An important feature of adding a second middle layer to a network, and not discussed in any of the relevant literature, is that this extra layer provides more consistency (although not necessarily any improvement) in a network's classification capabilities. This is true for both Phase and Power data. Since the second middle layer only slightly increases the number of iterations for Phase data (but sometimes does lead to training instability), and actually decreases the number for Power data, it would seem there is a good cost-performance trade-off in adding the second layer. In fact, it is almost a necessity for Power data.

One of the most disappointing results of this research is that no correlation can be obtained between middle layer size(s) and classification capabilities. At the same time, this might indicate that classification performance really is independent of layer size for this particular data. The fact that a relatively small number of middle layer neurodes are capable of classifying 60% or more of the patterns indicates that the *general* decision boundary is fairly simple, while the inability to classify more than 75% of the patterns could mean there is considerable class overlap that can only be resolved by adding more complexity to the general decision boundary by adding more neurodes in the middle layers.

The use of the multiple network BPN for classifying two different data types into the same class shows promise. This specialized network shows much more consistency in training and classification performance for a variety of network characteristics. A fault is that its subnetworks must be trained independently, but since training is performed off-line for BPNs, and the best performing networks may be chosen as the subnetworks, this problem may be minimized. At the very least, it provides some insight into a method for efficient anesthesia monitoring that allows multiple data types to determine patient status; it is conceivable that not only EEG, but EMG, oesophageal contractility, heart rate, etc, may be monitored at the same time by a multiple network BPN in order to provide the best determination of anesthetic depth (see Navabi [28] for an application of multiple NN's to an Operating Room smart alarm system).

Classification results for different values of θ were significant. There is definitely a trade-off in training performance vs. classification performance as θ is

increased. This contradicts Caudill's claim that the activation function constant is unimportant to network performance [5]. Typically, θ is simply set to "1" and left there, yet this research shows that significant improvements may be made, indeed may *have* to be made, by increasing or decreasing this value. The obvious consequence of increasing θ during classification is that the decision regions between classes become much sharper. This will tend to force a class choice for fuzzy patterns that are on the border between one class and another. Typically the choice is correct, but misclassification may also result. I must reassert that for the data used in this research, misclassification was virtually zero for all BPN testing. This fact is probably not true in general, but is extremely significant to using EEG monitoring for determining anesthetic depth.

One important training characteristic that affects classification results is the LMS Error criteria, ϵ . Since the majority of the networks used $\epsilon = 0.001$, most of the results discussed thus far have been for these networks. However, $\epsilon = 0.005$ and higher were also tried. Classification scores increase with $\epsilon = 0.005$, demonstrating that the choice of the LMS Error criteria is important to network performance. A higher value of ϵ not only decreases convergence time, but may increase classification score. This is true since a higher ϵ -value results in a more generalized network that should be able to better classify unfamiliar patterns. Increasing ϵ too high, however, results in a network that not only is unable to classify new patterns, but is also incapable of recognizing its own training set. It would appear from the results of this research that $\epsilon = 0.005$ is ideal for the data used. Again, misclassification was virtually zero for all testing. It is important that any research with BPNs designate the value of ϵ utilized, which is not the case for much of the specified literature.

The final, and possibly most important, feature of BPN classification is the classification performance criteria. None of the researchers in the specified literature discuss what their criteria were for deciding output neurode state (and, thus, pattern class) based on present neurode values. Yet this criteria, as demonstrated in this research, may provide a significant difference in classification performance.

It seems likely that using Criteria 2 (section 4.2.1.8) should increase misclassification. Once again I must emphasize that this was not the case for this research. Misclassification was zero.

4.3 SPSS Analysis

SPSS analysis on a VAX/VMS system was used in order to compare the BPN results with a statistical paradigm. Input data to the statistical analysis was identical to that used for the the BPNs.

4.3.1 Results

Discriminant analysis (DA) showed results as varied as those for the BPNs. Classification scores were dependent on the data type and the group used to form the discriminant. Tables 4.1–4.12 show the DA results. Power data was far more accurate than Phase data, as was true for the BPNs. Unlike the BPNs, however, Combined data agreed with Power rather than Phase data. Examining the manner in which the DA constructed its decision rules showed that for Combined data, discriminant analysis routinely rejected the Phase data (i.e. Phase data “failed the tolerance test”). Group2 also provided better results than Group1, as was true for the BPNs. Misclassification was as high as 40% for some of the data. DA was also at times unable to recognize patterns it had used in the analysis.

4.3.2 Comparisons with BPN Results

Misclassification is the most significant difference between the two methods. Because DA forces a pattern into a class, if the discriminants and decision rules are not completely representative of the given data, misclassification can result. BPN analysis, however, allows fuzzy patterns to be “unclassified”, i.e. output patterns such as [000] or [101]. It is more important for measuring depth of anesthesia that misclassifications be kept to a minimum. Therefore, BPNs are superior to DA for this application.

Table 4.13 shows comparisons between best BPN and DA performance. The first column denotes the type of data used in the analysis. The second column

Classification Results for Group Selected for Analysis

Actual Class	No. of Cases	Predicted Class Membership		
		MAC 0.5	MAC 1.0	MAC 1.5
MAC 0.5	5	5 100%	0 0%	0 0%
MAC 1.0	5	0 0%	5 100%	0 0%
MAC 1.5	5	0 0%	0 0%	5 100%

Table 4.1: SPSS Discriminant Analysis Phase Classification Results for Group1 Where Group1 Was Used in the Analysis to Form the Discriminant and Decision Rule.

Classification Results for Group Not Selected for Analysis

Actual Class	No. of Cases	Predicted Class Membership		
		MAC 0.5	MAC 1.0	MAC 1.5
MAC 0.5	5	3 60%	1 20%	1 20%
MAC 1.0	5	1 20%	4 80%	0 0%
MAC 1.5	5	1 20%	2 40%	2 40%

Table 4.2: SPSS Discriminant Analysis Phase Classification Results for Group2 Where Group1 Was Used in the Analysis to Form the Discriminant and Decision Rule.

Classification Results for Group Selected for Analysis

Actual Class	No. of Cases	Predicted Class Membership		
		MAC 0.5	MAC 1.0	MAC 1.5
MAC 0.5	5	4 80%	0 0%	1 20%
MAC 1.0	5	0 0%	5 100%	0 0%
MAC 1.5	5	0 0%	0 0%	5 100%

Table 4.3: SPSS Discriminant Analysis Phase Classification Results for Group2 Where Group2 Was Used in the Analysis to Form the Discriminant and Decision Rule.

Classification Results for Group Not Selected for Analysis

Actual Class	No. of Cases	Predicted Class Membership		
		MAC 0.5	MAC 1.0	MAC 1.5
MAC 0.5	5	0 0%	1 20%	4 80%
MAC 1.0	5	0 0%	2 40%	3 60%
MAC 1.5	5	0 0%	0 0%	5 100%

Table 4.4: SPSS Discriminant Analysis Phase Classification Results for Group1 Where Group2 Was Used in the Analysis to Form the Discriminant and Decision Rule.

Classification Results for Group Selected for Analysis

Actual Class	No. of Cases	Predicted Class Membership		
		MAC 0.5	MAC 1.0	MAC 1.5
MAC 0.5	5	5 100%	0 0%	0 0%
MAC 1.0	5	1 20%	4 80%	0 0%
MAC 1.5	5	0 0%	0 0%	5 100%

Table 4.5: SPSS Discriminant Analysis Power Classification Results for Group1 Where Group1 Was Used in the Analysis to Form the Discriminant and Decision Rule.

Classification Results for Group Not Selected for Analysis

Actual Class	No. of Cases	Predicted Class Membership		
		MAC 0.5	MAC 1.0	MAC 1.5
MAC 0.5	5	4 80%	1 20%	0 0%
MAC 1.0	5	2 40%	2 40%	1 20%
MAC 1.5	5	1 20%	2 40%	2 40%

Table 4.6: SPSS Discriminant Analysis Power Classification Results for Group2 Where Group1 Was Used in the Analysis to Form the Discriminant and Decision Rule.

Classification Results for Group Selected for Analysis

Actual Class	No. of Cases	Predicted Class Membership		
		MAC 0.5	MAC 1.0	MAC 1.5
MAC 0.5	5	5 100%	0 0%	0 0%
MAC 1.0	5	0 0%	5 100%	0 0%
MAC 1.5	5	0 0%	0 0%	5 100%

Table 4.7: SPSS Discriminant Analysis Power Classification Results for Group2 Where Group2 Was Used in the Analysis to Form the Discriminant and Decision Rule.

Classification Results for Group Not Selected for Analysis

Actual Class	No. of Cases	Predicted Class Membership		
		MAC 0.5	MAC 1.0	MAC 1.5
MAC 0.5	5	4 80%	0 0%	1 20%
MAC 1.0	5	1 20%	3 60%	1 20%
MAC 1.5	5	0 0%	0 0%	5 100%

Table 4.8: SPSS Discriminant Analysis Power Classification Results for Group1 Where Group2 Was Used in the Analysis to Form the Discriminant and Decision Rule.

Classification Results for Group Selected for Analysis

Actual Class	No. of Cases	Predicted Class Membership		
		MAC 0.5	MAC 1.0	MAC 1.5
MAC 0.5	5	5 100%	0 0%	0 0%
MAC 1.0	5	1 20%	4 80%	0 0%
MAC 1.5	5	0 0%	0 0%	5 100%

Table 4.9: SPSS Discriminant Analysis Combined Classification Results for Group1 Where Group1 Was Used in the Analysis to Form the Discriminant and Decision Rule.

Classification Results for Group Not Selected for Analysis

Actual Class	No. of Cases	Predicted Class Membership		
		MAC 0.5	MAC 1.0	MAC 1.5
MAC 0.5	5	4 80%	1 20%	0 0%
MAC 1.0	5	2 40%	2 40%	1 20%
MAC 1.5	5	1 20%	2 40%	2 40%

Table 4.10: SPSS Discriminant Analysis Combined Classification Results for Group2 Where Group1 Was Used in the Analysis to Form the Discriminant and Decision Rule.

Classification Results for Group Selected for Analysis

Actual Class	No. of Cases	Predicted Class Membership		
		MAC 0.5	MAC 1.0	MAC 1.5
MAC 0.5	5	5 100%	0 0%	0 0%
MAC 1.0	5	0 0%	5 100%	0 0%
MAC 1.5	5	0 0%	0 0%	5 100%

Table 4.11: SPSS Discriminant Analysis Combined Classification Results for Group2 Where Group2 Was Used in the Analysis to Form the Discriminant and Decision Rule.

Classification Results for Group Not Selected for Analysis

Actual Class	No. of Cases	Predicted Class Membership		
		MAC 0.5	MAC 1.0	MAC 1.5
MAC 0.5	5	4 80%	0 0%	1 20%
MAC 1.0	5	1 20%	3 60%	1 20%
MAC 1.5	5	0 0%	0 0%	5 100%

Table 4.12: SPSS Discriminant Analysis Combined Classification Results for Group1 Where Group2 Was Used in the Analysis to Form the Discriminant and Decision Rule.

gives the group number used to either form a discriminant or train a network, while the third column gives the group number used as the "unknown" patterns for classification. Columns 4 and 6 show the percent of Group1 patterns correctly classified and misclassified by either DA or BPN analysis, respectively. Finally, columns 5 and 7 show the percent of Group2 patterns correctly classified and misclassified by either DA or BPN analysis, respectively.

DA & BPN EEG Data Classification Results						
Data Type	Train Group	Test Group	DA		BPN	
			Classif.-Misclass. Group1	Group2	Classif.-Misclass. Group1	Group2
Power	1	2	93- 7%	53-47%	100- 0%	60- 0%
	2	1	80-20%	100- 0%	73- 0%	100- 0%
Phase	1	2	93- 7%	47-53%	100- 0%	53- 0%
	2	1	60-40%	100- 0%	60- 0%	100- 0%
Comb.	1	2	93- 7%	53-47%	100- 0%	53- 0%
	2	1	80-20%	100- 0%	60- 0%	100- 0%

Table 4.13: Comparison of EEG Data Classification and Misclassification Results for Discriminant and BPN Analysis.

Discriminant analysis has a 7% better classification score for Group1 Power and a 20% better score for Combined data over BPN analysis. In all other cases for Power, Phase and Combined data, BPN performance is equal or superior to DA. BPN results also show that, for the data used, recall for patterns previous presented is always 100%, while DA falls to 93% for Group1. The added importance of misclassification makes BPN analysis more appropriate for determining anesthetic depth.

5.0 CONCLUSIONS AND FUTURE STUDY

Performance of this Thesis satisfied the three research goals:

- A detailed account of NN training and testing characteristics based on various topologies and network parameters was presented. I was able to confirm training characteristics previously outlined by Rezgui *et al* [34] and Higashino *et al* [16]. I was also able to show that, contrary to most of the literature, the activation function constant is extremely important in training and classification, while layer size appears irrelevant to network capabilities *for the data sets used*.
- An optimal, feasible NN was proposed for each type of data used. A simple, one-layer BPN for Phase data demonstrated extremely fast convergence with a fair classification score. A two-layer BPN for Power data had slower convergence times, but better classification scores. Finally, a multiple network BPN demonstrated the ability to use optimized subnetworks in order to achieve faster convergence and better classification scores on combined data.
- The optimal NNs were compared with Discriminant Analysis methods. The NN was judged superior to DA because its comparable classification scores showed no misclassifications while DA had misclassifications from 20% to 47%.

The research results are considered valid for the following reasons: (1) Care was taken in accurately initializing networks to standard values; (2) There was agreement with mathematical and empirical results of relevant literature research; (3) Several thousand networks were used to collect data in order to create a statistically significant data base of results; (4) Statistical methods were used to compare and contrast results, providing an objective baseline; (5) Emphasis was placed on

the dependence of BPN training and classification results to data type; and (6) Training results were cross-checked with classification results, emphasizing that increasing the BPN training performance does not necessarily lead to an increase in classification performance.

Significantly, one of the most important aspects of this research is finding that different data types greatly affect the results. As stated in the Thesis introduction, results reported by researchers must specify data types in order to make valid assumptions and comparisons. It was originally encouraging to find that Phase data produced such excellent training results; This feeling was tempered by the disappointing classification results, particularly when compared with Power data. It would have been easy to only compare training results and state that Phase data was significantly better at determining anesthetic depth, an important “fact” previously unreported; however, truly meaningful research into BPNs (and Neural Networks in general) must emphasize training *and classification*. To the defense of Phase data, I feel that a larger EEG data base for training might provide the extra information needed to improve Phase-classification performance, while only slightly increasing training times.

The BPNs used in this research provided auxiliary information on the EEG data, which may be useful for future study. As a consequence of the training parameters, network topologies, and final weight magnitudes, several characteristics may be postulated about the EEG data. First, the Power data is much less homogeneous than the Phase data. Second, the Phase data has pattern boundaries that are much fuzzier than the Power data. Third, the Power data contains information from 10–20 Hz that, while considerably smaller in magnitude than data under 10 Hz, is, none-the-less, significant in determining pattern class. This information is important for finding what EEG data is truly useful to judging anesthetic depth.

Steps taken to optimize training rate, i.e. increasing η and α , are shown to have no effect on classification capabilities. Therefore, η and α may be increased to their maximum useful values for training without having to worry about effects on classification. Another parameter, θ , is shown to have a significant effect on both training and classification. It is important to restate the finding that increasing θ

during classification increases classification scores without increasing misclassifications. This has the benefit of providing a low θ for more stable training performance and then using a high θ for sharper classification boundaries.

Networks with two middle layers are shown to have more stable classification scores, a result previously unreported in the specified literature. They also provide optimal training for Power data.

Overall, BPNs adequately demonstrate their ability to accurately and feasibly determine anesthetic depth from intraoperative EEG data. Further, it can be postulated that an objective classification system that is able to maintain constant vigilance over an entire surgical procedure can be formed using BPNs on EEG data as well as on other forms of data.

Three main areas are considered important to future study: (1) Expanding the EEG data base; (2) Using EEG data in combination with other forms of physiological data; (3) Using other performance enhancement techniques in order to increase learning rate.

The size of the EEG data base used for this research was small when compared with other relevant research. This was unavoidable, but it is considered essential that the data base be expanded in order to ensure that the results be statistically significant for a larger number of subjects, and that the full classification capabilities (accuracies) be tested. As previously stated, this would be particularly beneficial to Phase training and testing. Effects of different anesthetic drugs on the EEG should also be used to expand the data base.

Using EEG data in combination with a variety of other physiological data is probably the best technique for determining anesthetic depth. Each form of data would serve as a backup and a verification for other forms. Monitoring most other forms of physiological data is less difficult than examining the electroencephalograph. The multiple network BPN demonstrated a feasible method for combining different data types into one classification system.

Finally, several researchers have proposed other performance enhancement techniques for BPNs in order to increase learning rate. If the EEG data base is expanded and used in combination with other forms of physiological data, there is

no doubt that methods will be needed to improve BPN convergence times. It is possible, in fact, that these performance enhancements may also improve results for the data sets used in this research. Further research in this area, however, should emphasize the *classification* capabilities of the networks as well as the training capabilities.

References

- [1] Bailey, David, and Donna Thompson. How to Develop Neural-Network, *AI Expert*, pp. 38–47, June 1990.
- [2] Berezowskyj, J.L. , J.A. McEwen, G.B. Anderson, and L.C. Jenkins. A Study of Anaesthesia Depth by Power Spectral Analysis of the Electroencephalogram (EEG), *Canadian Anaesthesia Society Journal*, vol 23, no. 1, pp. 1–8, January 1976.
- [3] Bershad, N.J. On the Optimum Gain Parameter in LMS Adaptation, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol ASSP-35, no. 7, pp. 1065–1068, July, 1987.
- [4] Cater, John P. Successfully Using Peak Learning Rates of 10 (and greater) in Back-propagation Networks with the Heuristic Learning Algorithm, *IEEE International Conference on Neural Networks*, vol 2, pp. 645–651, Washington, D.C., 1987.
- [5] Caudill, Maureen. *Neural Networks Primer*, Reprinted from issues of *AI Expert*, Miller Freeman Publications, San Francisco, 1989.
- [6] Caudill, Maureen. Neural Network Training Tips and Techniques, *AI Expert*, pp. 56–61, January 1991.
- [7] Chester, Daniel L. Why Two Hidden Layers Are Better Than One, *International Joint Conference on Neural Networks*, vol 1, pp. 265–268, Washington, D.C. , 1990.
- [8] Egbert, Dwight D. , Edward E. Rhodes, and Philip H. Goodman. Preprocessing of Biomedical Images for Neurocomputer Analysis, *IEEE International Conference on Neural Networks*, vol 1, pp. 561–568, 1988.
- [9] Gallinari, P. , et al. On the Relations Between Discriminant Analysis and Multilayer Perceptrons, *Neural Networks*, vol 4, pp. 349–360, 1991.
- [10] Gevins, Alan S. Neural Network Signal Processing in Human Brain Research, Part1: Methods, *IEEE Engineering in Medicine & Biology 10th Annual International Conference*, vol 3, 1988.
- [11] Gevins, Alan S. , and Nelson H. Morgan. Applications of Neural-Network (NN) Signal Processing in Brain Research, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol 36, no. 7, pp. 1152–1161, July 1988.
- [12] Gevins, Alan S. , Ronald K. Stone, and Scott D. Ragsdale. Differentiating the Effects of Three Benzodiazepines on Non-REM Sleep EEG Spectra, *Neuropsychobiology*, vol 19, pp. 108–115, 1988.

- [13] Gorman, R. , and T. Sejnowski. Analysis of Hidden Units in a Layered Network Trained to classify Sonar Targets, *Neural Networks*, vol 1, no. 1, pp. 75–89, 1988.
- [14] Hameroff, Stuart R. , Richard C. Watt, and Tim Jolly. EEG Monitoring for Anesthetic Depth Assessment, *IEEE Engineering in Medicine & Biology Society 10th Annual International Conference*, vol 4, pp. 1851–1852, 1988.
- [15] Hecht-Nielsen, Robert. Kolmogorov’s Mapping Neural Network Existence Theorem, *IEEE International Conference on Neural Networks*, vol 3, pp. 11–14, Washington, D.C., 1987.
- [16] Higashino, Junichi, Bart L. de Greef, and Eric H.J. Persoon. Numerical Analysis and Adaptation Method for Learning Rate of Back Propagation, *International Joint Conference on Neural Networks*, vol 1, pp. 627–630, Washington, D.C. , 1990.
- [17] Horne, Bill, and Don Hush. On the Optimality of the Sigmoid Perceptron, *International Joint Conference on Neural Networks*, vol 1, pp. 269–272, Washington D.C. , 1990.
- [18] Hudson, Robert J. , Donald R. Stanski, Lawrence J. Saidman, and Edward Meathe. A Model for Studying Depth of Anesthesia and Acute Tolerance to Thiopental, *Anesthesiology*, vol 59, pp. 301–308, 1983.
- [19] Izui, Yoshio, and Alex Pentland. Speeding Up Back Propagation, *International Joint Conference on Neural Networks*, vol 1, pp. 639–642, Washington, D.C. , 199.
- [20] Jordan, M.I. An introduction to Linear Algebra in Parallel Distributed Processing, in D.E. Rumelhart and J.L. McClelland (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol 1: Foundations*, MIT Press, Cambridge, Mass, 1986.
- [21] Kohonen, Teuvo. An Introduction to Neural Computing, *Neural Networks*, vol 1, no. 1, pp. 3–16, 1988.
- [22] Kreyszig, Erwin. *Advanced Engineering Mathematics*, John Wiley & Sons, Inc, New York, 1988.
- [23] Levy, Warren J. Power Spectrum Correlates of Changes in the Unconsciousness During Anesthetic Induction with Enflurane, *Anesthesiology*, vol 64, pp. 688–693, 1986.
- [24] Levy, Warren J. , Harvey M. Shapiro, Gary Maruchak, and Edward Meathe. Automated EEG Processing for Intraoperative Monitoring: A Comparison of Techniques, *Anesthesiology*, vol 53, pp. 223–236, 1980.

- [25] Lippman, Richard P. An Introduction to Computing with Neural Nets, *IEEE ASSP Magazine*, pp. 4–22, April 1987.
- [26] Lorentz, G.G. The 13-th Problem of Hilbert, in Felix E. Browder (ed), *Proceedings of Symposia in Pure Mathematics. Volume 28: Mathematical Developments Arising From Hilbert Problems*, American Mathematical Society, Providence, RI, 1976.
- [27] Meyer, Bernd, et al. Identification of the H-NMR Spectra of Complex Oligosaccharides with Artificial Neural Networks, *Science*, vol 251, pp. 542–544, February 1, 1991.
- [28] Navabi-Shirazi, Mohammed Jafar. *Integration of Operating Room Monitors for Development of a Smart Alarm System*, PhD Dissertation, Electrical & Computer Engineering Department, University of Arizona, 1990.
- [29] Norušis, Marija J. *SPSS Advanced Statistic's User's Guide*, SPSS, Inc, Chicago, 1990.
- [30] Oppenheim, Alan V. , and Jae S. Lim. The Importance of Phase in Signals, *Proceedings of the IEEE*, vol 69, no. 5, pp. 529–541, May 1981.
- [31] Oppenheim, Alan V. , and Alan S. Willsky. *Signals and Systems*, Prentice-Hall Signal Processing Series, Englewood Cliffs, NJ, 1983.
- [32] Pao, Yoh-Han. *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley Publishing Co, Inc, Reading, Mass, 1989.
- [33] Rayburn, D.B. , et al. Use of Backpropagation Neural Networks to identify Cardiovascular Waveforms, *IEEE INNS International Joint Conference on Neural Networks*, vol 2, pg. 105, 1990.
- [34] Rezgui, Ali, and Nazif Tepedelenlioglu. The Effect of the Slope of the Activation Function on the Back Propagation Algorithm, *International Joint Conference on Neural Networks 1990*, vol 1, pp. 707–710, Washington, D.C. , 1990.
- [35] Robson, J.G. Measurement of Depth of Anaesthesia, *British Journal of Anaesthesiology*, vol 41, pp. 785–788, 1969.
- [36] Rose, D.S. *Use of Fourier Analysis and Discriminant Function Analysis of the Electroencephalogram to Determine Anesthetic Depth*, Master's Degree Thesis, Electrical and Computer Engineering Department, University of Arizona, 1987.
- [37] Rumelhart, D.E. , G.E. Hinton and R.J. Williams. Learning Internal Representations by Error Propagation, in D.E. Rumelhart and J.L. McClelland (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol 1: Foundations*, MIT Press, Cambridge, Mass, 1986.

- [38] Schizas, C.N., C.S. Pattichis, I.S. Schofield, P.R. Fawcett, L.T. Middleton. Artificial Neural Net Algorithms in Classifying Electromyographic Signals, *First IEE International Conference on Artificial Neural Networks*, pp. 134–138, 1989.
- [39] Scott, James C., James E. Cooke, and Donald R. Stanski. Electroencephalographic Quantitation of Opioid Effect: Comparative Pharmacodynamics of Fentanyl and Sufentanil, *Anesthesiology*, vol 74, pp. 34–42, 1991.
- [40] Sebel, Peter S. Evaluation of Anaesthetic Depth, *British Journal of Hospital Medicine*, pp. 116–124, August 1987.
- [41] Stanley, William D., and Steven J. Peterson. Fast Fourier Transforms on Your Home Computer, *Byte*, pp. 14–22, December 1978.
- [42] Thomas, D.W., and W.B. Runciman. Monitoring Depth of Anaesthesia, *Anaesthesia and Intensive Care*, vol 16, no. 1, pp. 69–71, February 1988.
- [43] Von Lehmen, A., E.G. Paek, P.F. Liao, A. Marrakchi, J.S. Patel. Factors Influencing Learning by Backpropagation, *IEEE International Conference on Neural Networks*, vol 1, pp. 335–341, 1988.
- [44] Widrow, Bernard, & Samuel D. Sterns. *Adaptive Signal Processing*, Prentice-Hall Signal Processing Series, Englewood Cliffs, NJ, 1985.