

**GENERALIZATION CAPABILITY
OF FEEDFORWARD NEURAL NETWORKS
FOR PATTERN RECOGNITION TASKS**

A THESIS

*submitted for the award of the degree
of*

MASTER OF SCIENCE

by

NEEHARIKA ADABALA



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.**

AUGUST 1996

सर्वेन्द्रियगुणद्रष्ट्रे सर्वप्रत्ययहेतवे ।
असत्ताच्छाययोक्ताय सदाभासाय ते नमः ॥१४॥

sarvendriya-guṇa-draṣṭre
sarva-pratyaya-hetave
asatā cchāyayuktāya
sad-ābhāsāya te namaḥ

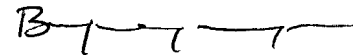
sarva-indriya-guṇa-draṣṭre—unto the seer of all objectives pursued by the senses; *sarva-pratyaya-hetave*—who is the solution to all doubts (and without whose help one cannot solve all doubts and inabilities); *asatā*—with the manifestation of unreality or illusion; *chāyayā*—because of the resemblance; *uktāya*—called; *sat*—of reality; *ābhāsāya*—unto the reflection; *te*—unto You; *namaḥ*—I offer my respectful obeisances.

TRANSLATION

My Lord, You are the observer of all the objectives of the senses. Without Your mercy, there is no possibility of solving the problem of doubts. The **material** world is **just** like a shadow resembling You. Indeed, one accepts this material world as real **because** it gives a glimpse of Your existence.

THESIS CERTIFICATE

This is to certify that the thesis entitled **Generalization Capability of Feedforward Neural Networks for Pattern Recognition Tasks** submitted by **Neeharika Adabala** to the Indian Institute of Technology, Madras for the award of the degree of Master of Science is a bona fide record of research work carried out by her under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.



(B. Yegnanarayana)

Madras 600 036

Date: *Aug 5, 1996*

To my parents

ACKNOWLEDGEMENTS

I gratefully thank my supervisor, Prof. B. Yegnanarayana, for his valuable guidance, and the time he spent in discussions with me despite his busy schedules. His dedication to work will always inspire me.

I thank my GTC members for their encouragement.

I thank, Mr. Manish Sarkar, for introducing me to the interesting field of fuzzy theory, and for the useful discussions I had with him.

I thank all the members of Speech and Vision Lab and Microprocessor Lab for the times they have helped me. I also thank my friends who have made my stay in Madras an enjoyable one.

ABSTRACT

One of the main advantages of Artificial Neural Networks (ANN) approach to **pattern recognition** task is the ability to **learn from** a few examples and generalize to give correct output for new examples. This **property**, known as generalization, is the focus of this work. There are various types of pattern recognition tasks. Generalization is possible in these pattern recognition tasks because of the presence of features in input data. In this thesis, we focus on the generalization capability of feedforward neural networks for pattern association tasks.

Although we expect good generalization performance of feedforward neural networks, it is not always possible to realize it. Analytical studies on the generalization studies make use of **a model of learning** from examples of synthetic data, which may not contain any features. Thus, the results obtained **have** limited application to many real world pattern recognition problems.

Practical **implementations** of learning from examples minimize certain objective functions in **order** to achieve better **generalization**. Methods to improve the generalization capability of neural networks involve **manipulation** of the parameters of the network so that better minimization of the objective function is achieved. The use of **variable** block sizes of data is suggested to improve the minimization of the objective function. But convergence to low value of the objective function does not always guarantee good generalization, as it may result in overtraining. To overcome this problem use of weight perturbation method and **an** alternative stopping criterion to error minimization are suggested. In all these methods, no **importance** is given to the features present in **the** data, and hence, there can only be a marginal improvement in the generalization. Apart from this, an objective function obtained from theoretical studies is used to evaluate the generalization capability of the network. An alternative method to quantify the generalization, called fuzzy generalization index, is proposed.

Generalization capability of a **neural** network can be significantly improved by approaches which incorporate knowledge about the problem into the neural network. Incorporation of knowledge into the network enables us to take care of the presence of features in the input data. The resulting generalization behavior is then comparable to the case of modeling of a system represented by data. This kind of behavior is present in some

extent in Radial Basis Function Neural Networks (RBFNN). The closeness property is incorporated, as knowledge about the problem, into the first layer of the RBFNN for classification tasks. When RBFNN is used for function approximation, the knowledge in the form of smoothness property of the function is incorporated into the network using a regularization term.

Knowledge about the problem can also be used for obtaining proper representation of data, to achieve good generalization. Here the data representation has to be chosen such that the network is able to capture the features present in the data rather than memorizing the data. This is illustrated by considering speech data. It is observed that the generalization performance is better when features that primarily represent the vocal tract system are used rather than when features that primarily represent the signal are used. Thus, the need to adopt a problem dependent approach to obtain good generalization in neural networks is brought out in this study.

CONTENTS

ACKNOWLEDGMENTS	v
ABSTRACT	vi
LIST OF ABBREVIATIONS	xi
LIST OF SYMBOLS	xi
LIST OF TABLES	xiii
LIST OF FIGURES	xiv
1 INTRODUCTION	1
1.1 Concept of Generalization	1
1.2 Generalization in Pattern Recognition Tasks	2
1.3 Generalization Capability of Neural Networks	3
1.4 Pattern Association by Neural Networks	3
1.4.1 Pattern Classification	5
1.4.2 Pattern Mapping/Function Approximation	5
1.5 Scope of this Thesis	6
1.6 Organization of the Thesis	7
2 GENERALIZATION: REVIEW OF SOME ANALYTICAL STUDIES	9
2.1 Introduction	9
2.2 Overview of some Measures of Generalization	10
2.2.1 Kullback-Leibler Measure	10
2.2.2 Cross-Validation Measure/Error Rate	11
2.2.3 Other Measures of Generalization	13

2.3	Model of Learning from Examples	13
2.4	Some Results from Theoretical Studies on Generalization	15
2.4.1	Results from Computational Learning	15
2.4.2	Theoretical Results on Asymptotic Behavior of Learning Curves . .	18
2.4.3	Discussion	19
2.5	Limitations of Theoretical Studies	20
2.6	Summary and Conclusions	21
3	GENERALIZATION IN FEEDFORWARD NEURAL NETWORKS	22
3.1	Introduction	22
3.2	Feedforward Neural Networks - Limitations in the Context of Generalization	24
3.3	Approaches to Improve Generalization	25
3.4	Suggested Methods for Improving Generalization	27
3.4.1	Stopping Criterion for Training	28
3.4.2	Variable Block Size Update Mode	30
3.4.3	Weight Perturbation with Training Examples	32
3.5	Quantification of Generalization	34
3.5.1	Issues in Measure of Generalization	35
3.5.2	Fuzzy Generalization Index	36
3.5.3	Results and Discussion	40
3.5.4	Limitations of Generalization Index	41
3.6	Summary	41
4	GENERALIZATION AS A PROBLEM DEPENDENT PHENOMENON	42
4.1	Introduction	42
4.2	Generalization in the Context of Specific Problems	43
4.3	Analogy with Modeling a System Represented by Data	43

4.4	Neural Networks with Problem-Specific Knowledge	48
4.4.1	Radial Basis Function Neural Networks for Pattern Classification .	48
4.4.2	Radial Basis Function Neural Networks for Pattern Mapping	50
4.5	Illustrations with Synthetic Data	50
4.5.1	Classification with Prior Knowledge	52
4.5.2	Function Approximation with Prior Knowledge	56
4.6	Summary	67
5	ILLUSTRATIONS OF GENERALIZATION STUDIES WITH SPEECH DATA	68
5.1	Introduction	68
5.2	Generalization in Vowel Recognition Task	69
5.2.1	The Vowel Classification Task	69
5.2.2	Results and Discussion	70
5.3	Generalization in Voice Conversion	71
5.4	Summary	72
6	SUMMARY AND CONCLUSIONS	74
6.1	Summary and Conclusions	74
6.2	Suggestions for Future Research	76
	APPENDIX	
A	Proof of Convergence to A <i>Posteriori</i> Class Probability	78
B	Overview of Some Fuzzy Set Concepts	80
B.1	Fuzzy Sets	80
B.2	Aggregation Operator	81
C	Experimental Observations for Generalization Index	82
	REFERENCES	85

LIST OF ABBREVIATIONS

ANN	Artificial Neural Network
BP	Backpropagation
FFNN	Feedforward Neural Network
HMM	Hidden Markov Model
LP	Linear Prediction
LPC	Linear Prediction Coefficients
MLP	Multilayer Perceptron
PAC	Probably Approximately Correct
RBF	Radial Basis Function
RBFNN	Radial Basis Function Neural Network
VC	Vapnik Chervonenkis

LIST OF SYMBOLS

$\Delta_H(i)$	The growth function of H .
$E[.]$	The expected value of a random variable.
Er	The generalization error.
Er_{emp}	The empirical estimate of Er by Cross Validation.
e_{kl}	The Kullback-Leibler measure.
$e_g(\mathbf{w}, k)$	The probability of misclassifying the $(k + 1)$ th example.
$e_g^*(\mathbf{w}, k)$	The entropic error.
\mathcal{F}	The family of functions a neural network can realize.
$f_{\mathbf{w}}$	The function realized by a neural network with weight \mathbf{w} .
G	The gain factor of a model.
\mathcal{G}	The fuzzy generalization index.
g_i	The generalization value attributed to the i th test samples.
$H(z)$	The transfer function of a model.
h_α	The fuzzy aggregation operator.
k	The number of training examples.
λ_i	The weights to the i th output node of a radial basis function neural network.

N	The set of natural numbers.
\mathcal{M}	The generalization measure.
$\mu_{correct}$	The fuzzy membership function that models the closeness of network output to desired output.
μ_{new}	The fuzzy membership function that models the newness of a test sample.
$\mu_{mol\ correct}$	The fuzzy membership function of <i>more or less</i> correct.
$\mu_{mol\ new}$	The fuzzy membership function of <i>more or less</i> new.
μ_i	The weights of the i th hidden node of a radial basis function neural network.
$Pr[.]$	The probability of an event.
p	The order of a linear prediction model.
p_m	The order of a linear prediction of the signal being modeled.
$\pi(\mathbf{x})$	The probability distribution of samples in input space.
$\pi(\mathbf{y} \mathbf{x})$	The class conditional probability.
$\pi(\mathbf{x}, \mathbf{y})$	The joint probability distribution.
\mathfrak{R}	The set of real numbers.
S	The input space.
S_n	The set of n test samples.
T_k	The set of k training examples.
$VCdim(H)$	The Vapnik-Chervonenkis dimension of H .
\mathbf{w}	$\mathbf{w} = [w_1\ w_2\ \dots\ w_n]^T$ weight vector of neural network.
\mathbf{w}_i	The weight vector associated with i th processor/node.
\mathbf{x}	$\mathbf{x} = [x_1\ x_2\ \dots\ x_n]^T$ is an n dimensional input to neural network.
\mathbf{x}_i	The i th input vector in the training set.
\mathbf{y}	$\mathbf{y} = [y_1\ y_2\ \dots\ y_m]^T$ is an m dimensional desired output of a neural network.
\mathbf{y}_i	The i th desired output vector in the training set.
\mathbf{Z}	The set of all possible input-output pairs.

LIST OF TABLES

1.1	Generalization in various pattern recognition tasks	6
3.1	Performance for alternative Stopping Criterion	30
3.2	Comparison between the fixed block and random block modes of update of weights	32
3.3	Performance of a neural network trained with weight perturbation by training examples.	34
4.1	Performance of an RBFNN trained on 25 examples for classification on syn- thetic 2-dimensional data.	54
4.2	Performance of an RBFNN trained on 100 examples for classification on syn- thetic 2-dimensional data.	55
4.3	Performance of MLP, with single hidden layer and, trained on 25 examples, for classification on synthetic 2-dimensional data.	55
4.4	Performance of MLP, with single hidden layer and, trained on 100 examples, for classification on synthetic 2-dimensional data.	56
4.5	Performance of the MLP with two layers of hidden nodes for classification on synthetic 2-dimensional data. The network is trained on 100 examples . . .	57
4.6	Summary of observations from Fig.4.8 to Fig.4.19 on comparison between MLP and RBFNN for function approximation.	59
5.1	Results of vowel classification using formant data	71
5.2	Results of vowel classification using cepstral data	72
C.1	Comparison of Generalization Index, \mathcal{G} , with (1-Error Rate) er for different test sets	83

LIST OF FIGURES

1.1	Typical Neural Network	4
1.2	Overview of the ideas described in this thesis	8
2.1	Shattering of a set of 3 points in 2D by straight lines	17
3.1	Typical feedforward neural network	23
3.2	Graph depicting <i>overtraining</i>	27
3.3	Illustration of <i>overfitting</i>	33
4.1	Function approximation with extra knowledge	44
4.2	Radial Basis Function Neural Network	49
4.3	Input space of an RBFNN as modeled by hidden layer	51
4.4	Input Space of Multilayer Perceptron as Modeled by Hidden Layer	52
4.5	The set of 2-dimensional data used to compare generalization capability of RBFNN and MLP for classification task	53
4.7	Function that is to be learned by neural networks	60
4.8	Training set of random samples of function corrupted by noise	60
4.9	Function realized by an MLP having 5 hidden nodes and trained on 50 examples	61
4.10	Function realized by an MLP having 10 hidden nodes and trained on 50 examples	61
4.11	Function realized by an MLP having 50 hidden nodes and trained on 50 examples	62
4.12	Function realized by an MLP having 100 hidden nodes and trained on 50 examples	62
4.13	Function realized by an MLP having 100 hidden nodes and trained on 100 examples	63
4.14	Function realized by an MLP having 150 hidden nodes and trained on 100 examples	63

4.15	Function realized by an RBFNN having 34 hidden nodes and trained on 100 examples. Smoothing parameter value is set to 0.3	64
4.16	Function realized by an RBFNN having 38 hidden nodes and trained on 100 examples. Smoothing parameter value is set to 0.6	64
4.17	Function realized by an RBFNN having 90 hidden nodes and trained on 100 examples. Smoothing parameter value is set to 1.0	65
4.18	Function realized by an RBFNN having 100 hidden nodes and trained on 200 examples. Smoothing parameter value is set to 0.3	65
4.19	Function realized by an RBFNN having 100 hidden nodes and trained on 200 examples. Smoothing parameter value is set to 0.6	66
4.20	Function realized by an RBFNN having 100 hidden nodes and trained on 200 examples. Smoothing parameter value is set to 1.0	66
C.1	Graph of variation of generalization error with increase in training examples .	84

Chapter 1

INTRODUCTION

1.1 Concept of Generalization

Generalization is an intuitive concept unique to human learning. For example, we learn the concept of addition of numbers by looking at several examples of addition along with some explanation provided by the teacher. Likewise, we learn the concept embedded in the written character, by observing and by writing several examples of the same character. We also learn the concept of a curve or a surface by observing several samples (sometimes noisy) of the points on the curve or the surface and with the additional knowledge of the class or classes of the objects to which the curve or surface belong. Thus learning from examples with additional knowledge many times forms the basis of the concept of generalization.

Generalization in learning from examples is possible because of some inherent features embedded in the input patterns or because of some constraints inherent in the mapping function. Learning and hence generalization, is not possible if we are presented with a set of random data **as** our examples. Thus, all problem situations are not generalizable.

Once we have learnt, that means we are capable of dealing with new situations such as a new addition problem or a new sample of character or a new point on the curve or surface. Our ability to deal with new situations can be evaluated by testing ourselves with several new examples, for which we know the answers for comparison. If our performance with this so called test data is better then we can say that our ability to generalize is better.

In most pattern recognition tasks, the performance of the pattern recognition system depends on its ability to generalize from training examples. Generalization concept is involved in all pattern recognition tasks, such **as** classification, mapping, storage, clustering, etc.

1.2 Generalization in Pattern Recognition Tasks

Pattern recognition has been an active area of research over the past few decades [6], [13]. Pattern recognition is a key element of many engineering solutions [4], e.g., speech recognition [43], image processing [53], [46], diagnostic application, seismic studies. Pattern recognition has been extensively studied because of this wide range of applicability. Pattern recognition tasks are broadly categorized into three different types [58], [19], namely:

1. **Clustering:** Data Clustering aims at discovering and emphasizing structure which is hidden in a data set [7], [28]. The structural relationships among individual data vectors are detected in an unsupervised method. In clustering the patterns of the same cluster possess some characteristics that make them exhibit some structure spatially. Generalization is possible here because new patterns have features similar to some known patterns, thus enabling the assignment to an existing cluster.
2. **Pattern Storage/Retrieval:** The task here is to store the patterns, and retrieve the corresponding original pattern when a partial/distorted pattern is given [19], [20]. This process by which an original pattern is recovered from a partial/distorted pattern can be viewed as generalization. Here it is possible to recover the original pattern because of features present in the given partial/distorted pattern.
3. **Pattern Association:** Here we associate with each pattern a corresponding response which may be interpreted as another pattern. Generalization is possible in these tasks because of presence of features in the data or because of certain characteristics in the mapping function between input and output patterns. Generalization for these kinds of problems forms the focus of this work.

1.3 Generalization Capability of Neural Networks

Various approaches to solve pattern recognition problems are investigated in literature [6], [53], [32]. Machines are still not able to perform pattern recognition tasks as efficiently as human beings do. This led to the development of the field of artificial neural networks that make use of mathematical models inspired by the functioning of the biologic l neural networks. An artificial neural network (ANN) can be defined as a system of

interconnected processors (p_i) or nodes [31]. The network has n inputs, which correspond to the elements of an n -dimensional input vector $\mathbf{x} \in \mathbb{R}^n$, and one or more output nodes. Each processor or node p , has associated with it a vector of weights w , having real valued elements. Various types of neural network architectures have been developed to solve different pattern recognition problems [58], [33]. Fig.1.1 illustrates a typical neural network with both feedback and feedforward connections. The ANN approach to pattern recognition problems has many advantages, namely, ability to deal with a wide variety of data (probabilistic, fuzzy and noisy data), fault tolerance, parallel processing, ability to learn from examples and '**generalization**'**capability** [25]. Most pattern recognition problems are too complex to solve entirely by handcrafted algorithms to take care of all the variations in data explicitly. Therefore, the ability of neural networks to learn from examples is a promising alternative [32]. Generalization capability of an ANN refers to its ability to learn from a few examples and give the desired output for samples that were not used in its learning phase [25]. Generalization is an important property in the context of learning from examples or modeling from examples, where we use a few examples to develop a model, that is able to give the desired output for patterns that were not used in the learning or modeling phase. This property is important because most pattern recognition tasks involve a variety of data, and it is not possible to train the neural network on all the examples.

ANNs offer promising results in a wide variety of recognition tasks. In this thesis we investigate the generalization capability of feedforward neural networks for pattern association tasks. In the following section we describe various kinds of pattern association tasks.

1.4 Pattern Association by Neural Networks

Pattern association problems involve association of input vectors with output. vectors. The neural networks that are used to solve the pattern association problems capture the input-output relation. Pattern association problems are of two major types:

1. Problems where it is possible to list all input-output pairs, for example, XOR problem.

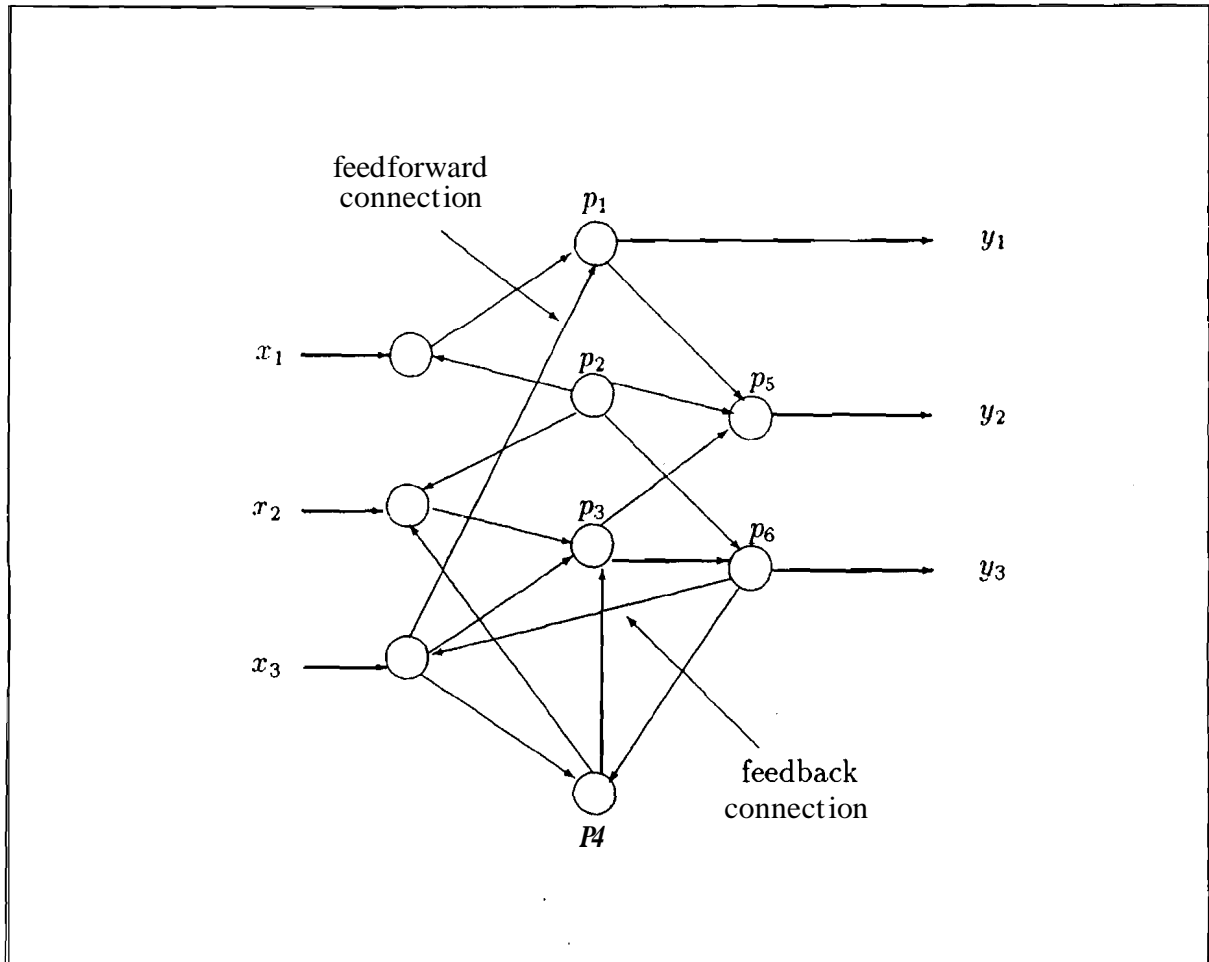


Figure 1.1: Typical neural network: with three inputs x_1 to x_3 and three outputs y_1 to y_3 . The processors or nodes are denoted by p_1 to p_6 .

2. Problems where it is not possible to list all the input-output pairs, for example, a large dimension parity problem.

In the first case all the patterns and the corresponding outputs can be stored and recalled, whereas it is not possible to do so in the second case. In the latter case, we require the neural network to give the desired output for inputs, without actually learning all possible input-output pairs. The network should capture the desired characteristics of all inputs from a few examples so that it can perform the desired association, i.e., it should generalize.

It is not always possible to generalize from a few examples [25]. Some examples of the problems where it is not possible to generalize are:

- An unsmooth mapping function where there is no apparent method to interpolate between given examples of the function.
- A large dimension parity problem where the feature is deep and hidden due to many surface features.
- Input-output associations like names and numbers in the telephone directory where there is no common property to capture from examples in the training set and generalize to yield the desired output for examples not present in the training set.

There exist various types of pattern association problems, where generalization is essential. In this thesis we focus on two types of pattern association tasks, namely, pattern classification and pattern mapping/function approximation, where generalization is possible. These tasks are described in the following two subsections.

1.4.1 Pattern Classification

In this type of pattern association problems we learn a many-to-one mapping from a training set consisting of input-output pairs. The grouping of inputs is done based on some underlying similarity in the example. Here generalization is possible because of the common features in the patterns of the same class which are transformed into proximity in some feature space.

1.4.2 Pattern Mapping/Function Approximation

In these problems, we are given a training set consisting of examples of an unknown function $f(\mathbf{x})$ in the form of input values of \mathbf{x} and corresponding output y . A pattern association system is required to approximate this function based on the training set, such that it gives the desired output for samples not present in the training set. That is, generalize from a few examples to find the desired function $f(\mathbf{x})$. This is possible if the function $f(\mathbf{x})$ exhibits smoothness. It should be noted that, the roughness of the function being approximated determines the scope for generalization.

Table 1.1 summarizes the properties which enable neural networks to generalize in the context of various pattern recognition tasks.

<u>Pattern Recognition Tasks</u>	<u>Reason why Generalization is Possible</u>
1. Pattern Clustering	Common structure in data belonging to same cluster.
2. Pattern Storage/Retrieval	Features in the partial/distorted pattern.
3. Pattern Association	
<ul style="list-style-type: none"> • Tasks where generalization is not possible 	Random associations with no features involved or tasks where the features are deeply hidden.
<ul style="list-style-type: none"> • Tasks where generalization is possible 	
<ul style="list-style-type: none"> – Pattern Classification 	Similar features in the patterns of same class.
<ul style="list-style-type: none"> – Pattern Mapping/Function Approximation 	Smoothness in the mapping function.

Table 1.1: Properties which enable neural networks to generalize in the context of various pattern recognition tasks.

1.5 Scope of this Thesis

In this thesis we address the issues of generalization in feedforward neural networks for pattern recognition tasks. First a conceptual understanding of generalization in the context of various pattern recognition tasks is provided.

The property of generalization is discussed from the point of view of learning from examples. The model of learning from examples is discussed along with results of some analytical studies on generalization. Limitations of the current implementations of learning from examples with respect to the issue of generalization are discussed.

Some methods of improving generalization capability are studied. Limitations of problem-independent methods for improving generalization are discussed. It is shown that an intuitively good generalization can be obtained only if knowledge of the problem is incorporated into the network. Methods for improving generalization in pattern mapping and pattern classification tasks are presented. The results are discussed with special reference to problems in speech recognition.

Fig.1.2 gives a brief overview of the work presented in this thesis.

1.6 Organization of the Thesis

In this chapter we have presented the conceptual understanding of generalization in the case of pattern recognition tasks.

In chapter 2 we present the theoretical studies on generalization. A model of learning from examples and some theoretical results are reviewed. Limitations in practical application of the theoretical studies on generalization are discussed.

In chapter 3 we propose problem-independent methods for improving generalization capability of neural networks. A new method of quantifying generalization capability that makes use of fuzzy set theory is proposed. Limitations of the suggested method are discussed.

In chapter 4 the problem dependence of the generalization phenomenon is described. The desired generalization behavior of neural networks is discussed by considering an analogy with modeling of a system represented by data. We focus on the pattern mapping and pattern classification problems.

Chapter 5 illustrates studies on problem-dependence of generalization with examples from speech recognition problems.

We summarize the work in chapter 6 and propose suggestions for future work.

Generalization in Feedforward Neural Networks for Pattern Recognition Tasks

1. Concept of Generalization in Pattern Recognition Tasks: Clustering, Pattern Storage/Retrieval and Pattern Association
2. Analytical Studies: Computational learning and Statistical approach
 - Consider synthetic data and artificial pattern associations
3. Generalization in Feedforward Neural Networks:
 - Problem independent approaches to improve generalization:
 - Multiple block update mode
 - Weight perturbation approach
 - Alternative stopping criterion
 - Alternative method to quantify generalization
 - Fuzzy generalization index
4. Problem Dependence of Generalization:
 - Analogy with modeling of a system represented by data
 - Problem-dependent methods of improvement of generalization
 - *Pattern Classification*: Radial Basis Functions for classification
 - *Pattern Mapping*: Regularization
5. Application to Speech:
 - *Pattern Classification*: Vowel classification
 - *Pattern Mapping*: Voice Conversion

Figure 1.2: Overview of the ideas described in this thesis.

Chapter 2

GENERALIZATION: REVIEW OF SOME ANALYTICAL STUDIES

2.1 Introduction

In this chapter some analytical studies on the property of generalization are reviewed. Although a relatively small fraction of the overall work done on neural networks is on theoretical analysis of generalization, these studies are marked by a variety of approaches. Some of the significant approaches are the computational learning theory approach [16], [5] and the statistical approach [27], [49]. These studies consider synthetic models of pattern recognition tasks and analyze the behavior of learning models for such tasks. The learning models optimize certain objective criteria formulated with respect to the synthetic data. The studies on generalization try to predict the generalization performance of the learning model. In this chapter we review some analytical results on generalization which give us an idea of the factors on which generalization capability depends in the existing approaches of learning from examples.

In section 2.2 we review some measures of generalization which are necessary for studies on generalization. A model of learning from examples is presented in section 2.3. Theoretical results obtained from computational learning theory approach and statistical theory approach are reviewed in section 2.4. A discussion on these results and their limitations, including reasons for the lack of direct applicability of the results to real world pattern recognition tasks is presented in section 2.5. Section 2.6 gives a summary of this chapter.

2.2 Overview of some Measures of Generalization

In order to study generalization capability we should be able to quantify it, that is, it should be possible to evaluate a network and decide when its generalization is 'good'. However, the notion of 'good' or 'reasonable' is itself not well defined. It varies from person to person and is problem dependent. For example, when the desired output is obtained on most occasions it is considered as 'good' generalization certain times, while in certain other types of problems generalization is considered to be 'good' if the network gives the desired output for a very rare situation which never occurred before. Various methods of measuring generalization are used in practice [34], [38].

In this section we give a brief overview of some of the measures that are used to quantify generalization. The subsection 2.2.1 is on Kullback-Leibler Measure, which is difficult to implement as it requires prior knowledge about the actual function being realized. In subsection 2.2.2, the Cross-Validation measure is described which can be implemented in practice but is computationally expensive. The generalization measures that are used in theoretical studies are presented in the section 2.2.3.

2.2.1 Kullback-Leibler Measure

The measure discussed in this subsection is defined for networks used for classification purpose. When a neural network is analyzed as a classifier we can view it as a probabilistic model which captures the probabilistic behavior of the system that generates the examples used for training. From this point of view, a measure of generalization is defined which measures the difference between the actual probabilistic system parameters and the system obtained by training the neural networks on the training examples [18]. This method of measuring generalization is called *Kullback-Leibler* measure and is defined as follows:

The class conditional probability distribution of the sample space is given by $\pi(\mathbf{y}|\mathbf{x})$, the probability distribution of \mathbf{x} is $\pi(\mathbf{x})$ and the joint probability distribution is $\pi(\mathbf{x}, \mathbf{y}) = \pi(\mathbf{y}|\mathbf{x})\pi(\mathbf{x})$. It is assumed that \mathbf{y} given \mathbf{x} is *identically independently distributed (i.i.d)*. The function approximated by the neural network, after training, is given by $f_{\mathbf{w}}(\mathbf{y}|\mathbf{x})$, where \mathbf{w} is the weight vector of the neural network that is available to adjust and minimize the error function according to the training set $T_k = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_k, \mathbf{y}_k)\}$ consisting of k examples. The Kullback-Leibler measure gives the distance between the two

functions $\pi(\mathbf{y}|\mathbf{x})$ and $f_{\mathbf{w}}(\mathbf{y}|\mathbf{x})$. This can be exploited as a measure of generalization error because it gives how well the actual system generating the examples is approximated. Mathematically, the Kullback-Leibler measure (e_{kl}) is given by the following equation:

$$e_{kl} = - \int \pi(\mathbf{x}, \mathbf{y}) \log \left[\frac{f_{\mathbf{w}}(\mathbf{y}|\mathbf{x})\pi(\mathbf{x})}{\pi(\mathbf{y}|\mathbf{x})\pi(\mathbf{x})} \right] d\mathbf{x}d\mathbf{y} \quad (2.1)$$

where the integral is over the input-output space. The equation (2.1) can be written as.

$$e_{kl} = -E [\log (f_{\mathbf{w}}(\mathbf{y}|\mathbf{x}))] + E [\log(\pi(\mathbf{y}|\mathbf{x}))] \quad (2.2)$$

in which the expectation is with respect to (\mathbf{x}, \mathbf{y}) .

The value of e_{kl} is equal to zero when the function approximated by the neural network is equal to the actual function, i.e., $f_{\mathbf{w}}(\mathbf{y}|\mathbf{x}) = \pi(\mathbf{y}|\mathbf{x})$. As the weights \mathbf{w} are adjusted according to a training set T_k which has examples selected at random, the term $-E[\log(f_{\mathbf{w}}(\mathbf{y}|\mathbf{x}))]$ is a random variable. Therefore, the first term in equation (2.2) can be used to define the generalization error (Er) as follows:

$$Er = -E [\log(f_{\mathbf{w}}(\mathbf{y}|\mathbf{x}))] \quad (2.3)$$

with expectation taken over (\mathbf{x}, \mathbf{y}) in T_k .

The Kullback-Leibler measure requires us to know the underlying probability distribution $\pi(\mathbf{x}, \mathbf{y})$, which is unknown in most cases. Therefore an alternative method of measuring generalization is needed. One such measure, the cross-validation measure is discussed in the following subsection.

2.2.2 Cross-Validation Measure/Error Rate

Cross-validation is a method of estimating the generalization error by making use of the training/test data [34]. In this method, generalization error, Er, given by equation (2.3), can be estimated as follows:

$$Er_{emp}(T_k, |\mathbf{w}|) = -\frac{1}{k} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in T_k} \log(f_{\mathbf{w}_{-i}}(\mathbf{y}_i|\mathbf{x}_i)) \quad (2.4)$$

Notation Er_{emp} is used because it gives an empirical estimate of Er. In the equation (2.4), \mathbf{w}_{-i} is the weight vector obtained by learning a training set T_k with its i th sample

deleted. Here $|\mathbf{w}|$ indicates number of adjustable parameters, i.e., weights $|\mathbf{w}| \in \mathbf{M}$ where \mathbf{M} is a set of all models under consideration. The weight vector \mathbf{w}_{-i} is called *jack knife* estimator. The above formula gives an estimate of the average predictive error on all examples, and $-\log(f_{\mathbf{w}_{-i}}(\mathbf{y}_i|\mathbf{x}_i))$ is the estimate of the predictive error on the i th sample by the rest of the samples in the training set, which are used to calculate the optimal weights \mathbf{w}_{-i} . It can be shown that Er_{emp} is an unbiased estimate of the generalization error Er as follows:

$$E[Er_{emp}(T_k, |\mathbf{w}|)] = -\frac{1}{k} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in T_k} E[\log(f_{\mathbf{w}_{-i}}(\mathbf{y}_i|\mathbf{x}_i))] \quad (2.5)$$

$$= -E[E(\log(f_{\mathbf{w}_{-i}}(\mathbf{y}_i|\mathbf{x}_i)))] \quad (2.6)$$

$$\approx -E[Er_{emp}(T_k, |\mathbf{w}|)] = Er \quad (2.7)$$

In equation (2.5) the expectation is taken with respect to (\mathbf{x}, \mathbf{y}) in T_k . Equation (2.7) is obtained by using equation (2.4) and equation (2.3). For test data $S_n = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \dots, n\}$ we have the expectation of error on test data set as,

$$E\left[-\frac{1}{n} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in S_n} \log(f_{\mathbf{w}}(\mathbf{y}_i|\mathbf{x}_i))\right] = -E[\log(f_{\mathbf{w}}(\mathbf{y}|\mathbf{x}))] = E[Er_{emp}(T_k, |\mathbf{w}|)] \quad (2.8)$$

which is the expectation taken with respect to (\mathbf{x}, \mathbf{y}) in T_k, S_n . Thus, Er_{emp} is also an unbiased estimate of the expectation of the error on the test data set. Due to these properties we can use Er_{emp} as an unbiased estimator of the generalization error.

But this method of cross validation to estimate generalization error involves training the network again and again by deleting different data examples from the training set. This is a time consuming and a computationally expensive procedure.

The most commonly used measure of generalization for pattern classification task is the percentage misclassification of the test samples or the **error rate** measure. It is assumed that the test samples are not used for the training of the network. This measure is extensively used because it is simple and easy to implement. It can be viewed as a variation of the cross validation measure.

In the following subsection we define some of the measures of generalization that are used in theoretical studies.

2.2.3 Other Measures of Generalization

Generalization error can be measured by considering it as the probability that the network trained on k examples of the training set gives the output for the $(k + 1)$ th sample incorrectly [2], [23]. If we represent this error by $e_g(\mathbf{w}, k)$ then it is given by the following equation:

$$e_g(\mathbf{w}, k) = Pr[\{f_{\mathbf{w}}(\mathbf{x}_{k+1}) \neq \mathbf{y}_{k+1}\}] \quad (2.9)$$

where $f_{\mathbf{w}}$ is the function output calculated by the neural network with weights equal to \mathbf{w} . When the probability of misclassification is high the generalization error value given by the above equation is high and vice versa.

Another method of generalization measure is to consider the **entropic** error or **entropic** loss [26]. It is defined as the negative logarithm of the probability of correct classification of $(k + 1)$ th pattern. That is, if entropic error is denoted by $e_g^*(\mathbf{w}, k)$ then,

$$\begin{aligned} e_g^*(\mathbf{w}, k) &= -\log(\text{Probability of correct classification}) \\ &= -\log(1 - \text{Probability of wrong classification}) \\ &= -\log(1 - e_g(\mathbf{w}, k)) \end{aligned} \quad (2.10)$$

It is clear that when the probability of correct classification is one, the value of the entropic error is zero.

As methods to quantify generalization are known, we give a learning model in the following section which abstracts the process of learning and makes it possible to theoretically study the concept of generalization. The model of learning from examples makes it possible to theoretically study the process of learning by using some of the measures discussed in this section.

2.3 Model of Learning from Examples

Learning from examples is a complex process which is not easy to understand. In order to analyze it, a model of learning from examples is used. One of the main results of learning from examples is the ability to generalize to give desired output for examples not used for learning. Thus, studies of generalization make use of a model of learning, and the

key idea is to compute the probability that the network gives the correct output for new samples after learning from a training set.

The concept of learning from examples is modeled through three components:

1. A system which generates random vectors on some fixed unknown probability distribution $\pi(\mathbf{x})$.
2. A supervisor that returns an output vector \mathbf{y} for every input vector \mathbf{x} according to a conditional probability distribution function $\pi(\mathbf{y}|\mathbf{x})$, which is also fixed but unknown. This includes the case when the supervisor uses a function $\mathbf{y} = f(\mathbf{x})$.
3. A learning machine capable of implementing a set of functions $f(\mathbf{x}, \mathbf{w})$, $\mathbf{w} \in \Lambda$, where Λ is the parameter space of learning machine.

Learning consists of selecting a function from a set of functions the learning machine can implement, such that the response of the machine is similar to the supervisor's response. This selection is done by using a training set of k *i.i.d* observations $T_k = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_k, \mathbf{y}_k)\}$ drawn according to $\pi(\mathbf{x}, \mathbf{y}) = \pi(\mathbf{x})\pi(\mathbf{y}|\mathbf{x})$.

In order to choose the best approximation to the supervisor's response the difference between the output of the learning machine and the supervisor is minimized. Let $\pi(\mathbf{z})$ represent the probability distribution on input-output space \mathbf{Z} . Consider a set of functions $Q(\mathbf{z}, \mathbf{w})$, $\mathbf{w} \in \Lambda$. The goal is to minimize the risk functional,

$$R(\mathbf{w}) = \int Q(\mathbf{z}, \mathbf{w}) d\pi(\mathbf{z}) \quad \mathbf{w} \in \Lambda \quad (2.11)$$

If the probability measure $\pi(\mathbf{z})$ is unknown, the minimization is done on a set of *i.i.d* examples $\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_k$ where \mathbf{z} is an example from the input-output space, i.e., (\mathbf{x}, \mathbf{y}) .

All learning problems are particular cases of this general problem of minimizing the risk functional on the basis of empirical data. Learning theory addresses the following issues with regard to the model of learning from examples given above,

1. Consistency of learning process.
2. Rate of convergence of learning process.
3. Generalization ability of learning process.

4. Development of learning algorithms.

Details about issues 1, 2 and 4 can be found in [55] and references thereof. In this work we concentrate on the third issue. Variations of the model of learning from examples presented in this section are analyzed to obtain results on generalization. Some analytical results on generalization are reviewed in the following section.

2.4 Some Results from Theoretical Studies on Generalization

At present there is no single complete theory of generalization, because the interpretations given to the functioning of the neural network vary. In the following two subsections we state results obtained from theoretical studies of networks using different approaches of analyses.

2.4.1 Results from Computational Learning

The discussion in this section makes use of the following intuitive idea of generalization [21]: Consider a network which has been satisfactorily trained using a sequence of training examples from a particular problem. If there is a 'high enough' probability that the actual error of the network for future samples drawn from the same problem is 'small enough' then we say that the network generalizes.

This idea of the concept of generalization is used in the Probably Approximately Correct (PAC) learning theory [16], which is based on the learning model introduced by Valiant [54]. In this section we define some terms that are essential to understand the theoretical results obtained in PAC theory. We give the definitions that follow in the context of neural networks for easier understanding. In the following definitions, \mathcal{F} denotes the class of functions that can be implemented by a neural network, and $f_{\mathbf{w}}$ represents one of the members of this class for a particular value of weight vector \mathbf{w} . \mathbf{S} is the input space.

Definition 1: (Dichotomy) Given a finite set $\mathbf{S} \subseteq \mathbb{R}^n$ and some function $f_{\mathbf{w}} \in \mathcal{F}$, we define the dichotomy (S^+, S^-) of \mathbf{S} , where S^+ and S^- are disjoint subsets of \mathbf{S} , such that, $S^+ \cup S^- = \mathbf{S}$ and $\mathbf{x} \in S^+$ if $f_{\mathbf{w}}(\mathbf{x}) = 1$, whereas, $\mathbf{x} \in S^-$ if $f_{\mathbf{w}} = 0$.

Definition 2: The hypothesis $h_{\mathbf{w}}$ associated with a function $f_{\mathbf{w}}$ is the subset of \mathbb{R}^n for which $f_{\mathbf{w}}(\mathbf{x})=1$, that is,

$$h_{\mathbf{w}} = \{\mathbf{x} \in \mathbb{R}^n | f_{\mathbf{w}}(\mathbf{x}) = 1\} \quad (2.12)$$

The hypothesis space H computed by the neural network is the set given by,

$$H = \{h_{\mathbf{w}} | \mathbf{w} \in \mathbb{R}^{|\mathbf{w}|}\} \quad (2.13)$$

It is the set of all hypothesis, where $|\mathbf{w}|$ is the total number of weights in the network.

Definition 3: Given a hypothesis space H and a finite set $S \subseteq \mathbb{R}^n$, we define $\Delta_H(S)$ as the set,

$$\Delta_H(S) = \{h_{\mathbf{w}} \cap S | h_{\mathbf{w}} \in H\} \quad (2.14)$$

We say that S is shattered by H , if $\Delta_H(S) = 2^{|S|}$ where $|S|$ is the number of elements of the set S .

Definition 4: (Growth **Function**) The growth function, $\Delta_H(i)$, is defined on the set of positive integers as,

$$\Delta_H(i) = \max_{S \subseteq \mathbb{R}^n, |S|=i} (|\Delta_H(S)|) \quad (2.15)$$

The growth function gives the maximum number of distinct dichotomies induced by H for any set of i points.

Definition 5: (Vapnik-Chervonenkis Dimension) The Vapnik-Chervonenkis dimension or VC dimension of the hypothesis space H , denoted by $VCdim(H)$, is the largest integer i such that $\Delta_H(i) = 2^i$. In the case when no such i exists $VCdim(H)$ is infinity.

A hypothesis space, H , is directly related to a class of functions, \mathcal{Z} , so we can apply the definitions of growth function and VC dimension to \mathcal{Z} .

Fig. 2.1 illustrates the shattering of **3** noncolinear points by straight lines. A set of **3** noncolinear points is the largest set of points that can be shattered in 2 dimensional space by straight lines, therefore the VC dimension of the set of straight lines with respect to a set of noncolinear points in 2 dimensional space is **3**.

VC dimension is a combinatorial parameter which measures the expressive power of the network. VC dimension has been used extensively to obtain results that tell us about

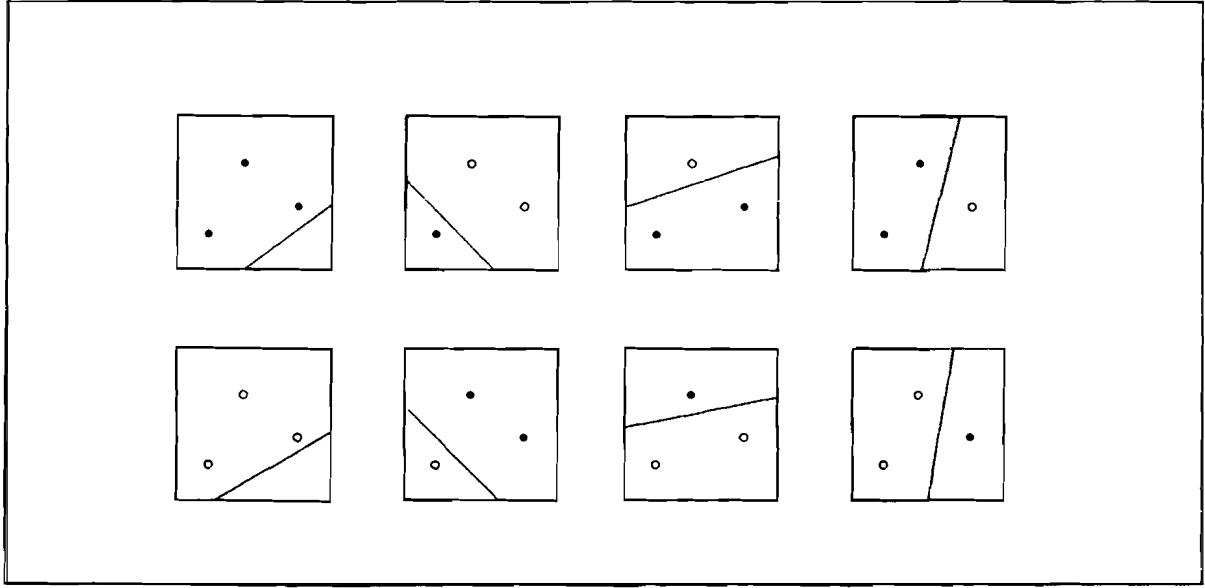


Figure 2.1: Shattering of three noncolinear points by straight lines. Thus VC dimension is three for straight lines in two dimensional space on a set of noncolinear points [25]

the ability of a classifier to generalize after it has been trained [5], [50], [3]. It has been shown [5] that it is not the size of the set of computable functions but the VC dimension of the functions that is crucial for good generalization, in the context of PAC learning model [5]. The results that follow use probabilistic definition of generalization error, given by equation (2.8), and give the worst case bounds of this error. The following key results on the bounds of the generalization error are given by Haussler, *et al.* [17]:

$$E[e_g(\mathbf{w}, k)] \leq \frac{VCdim(\mathcal{L})}{k+1} \quad (2.16)$$

where E is the expectation, \mathcal{L} is the class of target functions and k is the number of training patterns.

The bound given in the equation (2.16) has been further improved and is given by

$$E[e_g(\mathbf{w}, k)] \leq \frac{1}{2} \frac{VCdim(\mathcal{L})}{k+1} \quad (2.17)$$

The results given by equation (2.16) and equation (2.17) are tighter bounds on the generalization error than the more powerful results obtained previously by Haussler *et al.*

[15] where fewer assumptions are made. It should be noted that the results given above are obtained by assuming that the neural networks performance is optimum when it implements a *Bayes Optimal Classification algorithm* [13]. The Bayes optimal classification algorithm makes use of finite training set to give an optimal prediction for a new sample. and it is different from a Bayes classifier that requires complete statistics of a classification problem. The use of \mathcal{L} , a class of target functions, models the fact that the classifier can be applied to a selection of different problems.

These results have been compared with the performance of neural networks on a classification task and the second bound has been found to be a moderately good approximation of the worst case generalization error [22]. The experiments conducted by Holden and Niranjana [22] were performed on Peterson/Barney Data which is real data unlike the synthetic data which is considered for analysis.

The main problem with the results given above is that the VC dimension of the function that we are approximating, is required. But this function is unknown. Consequently, the value of the bound cannot be calculated correctly. Apart from this, the calculation of VC dimension of various classes of functions is not easy. VC dimensions of some classes of functions are given in [2], [50], [57].

2.4.2 Theoretical Results on Asymptotic Behavior of Learning Curves

When the performance of the neural network is plotted against the training patterns then the resulting curve is known as a learning curve. The learning curve shows how quickly a learning network improves behavior that is evaluated by the generalization error [27]. Thus, study of the behavior of the curves gives us an idea about the generalization capability.

A universal result on the behavior of the entropic error $e_g^*(\mathbf{w}, k)$ with increase in the training examples in the training set is given by Amari [26]. According to this result, when every weight of a neural network is contributing to the performance of the neural network, that is under the condition of regularity, the entropic learning curve is asymptotically given

by,

$$\langle e_g^*(\mathbf{w}, b) \rangle \sim \frac{|\mathbf{w}|}{k} \quad (2.18)$$

where $|\mathbf{w}|$ stands for number of weights, k is the number of patterns the network is trained on and $\langle e_g^*(\mathbf{w}, b) \rangle$ indicates the average over all training with different training sets. It should be noted that this result is independent of the architecture of the neural network and the learning algorithm used to train it.

2.4.3 Discussion

In the above two subsections we gave two results on the behavior of generalization error. In this section we bring out some similarities in the results which were obtained from different approaches of analysis.

The VC dimension of a network can be regarded as a measure of capacity or expressive power of a neural network. The number of weights also indicates the capacity of a neural network. So both the results show that the generalization error is directly proportional to the capacity of the network. Moreover, both results show that the generalization error is inversely proportional to the number of patterns used to train the neural network.

It has been shown that, in case of Radial Basis Functions Neural Networks (RBFNN), $|\mathbf{w}| - 1 \leq VCdim(\mathcal{F}) \leq |\mathbf{w}|$, where \mathcal{F} is the family of functions that a network can approximate and $|\mathbf{w}|$ is the number of weights [2]. In the case of polynomial basis networks, $VCdim(\mathcal{F}) = |\mathbf{w}|$. Based on the above results on VC dimension one can see that, the bounds of the generalization error obtained from computational learning and the behavior of the learning curves essentially give similar results. But, in computational learning the worst case behavior of the error is studied while in the latter case the average behavior is analyzed [21].

Relationship between these theoretical methods are discussed by Sueng et al. [49]. In the following sections we present the limitations of the theoretical studies with respect to real world pattern recognition problems.

2.5 Limitations of Theoretical Studies

As generalization capability of a neural network is an important property, it is useful to have analytical results which can be utilized as tools in the design of neural networks for practical applications. But the analytical results presented in this chapter are not directly applicable for this purpose because of the following reasons:

- They are obtained by considering synthetic data generated by a model or by considering random data associations that do not model actual pattern recognition problems which exhibit some features that can be generalized in them. All bounds, on the number of training examples needed to guarantee good generalization are found to be large compared to the number of examples that are usually required in practice. This is mainly because random associations of data are also considered while finding the bounds. Apart from this, theoretical studies usually make an assumption of noise free data which is an unlikely situation in real world pattern recognition problems.
- The learning process is tailored to the synthetic data. Usually such data does not exhibit features. Consequently, the objective functions which are minimized during the selection of the parameters for the learning machine are not designed to capture the features in the data, that enable generalization.
- One of the key assumptions, that is made in analytical studies of generalization in neural network, is that every weight contributes to the approximation of the function. But when large networks are considered, one cannot ensure that every weight is contributing to the function approximation. Some weights may have values that negate the effect of each other during the calculation of the output, so they just balance each other, and thus do not contribute to the function approximation. Such weights contribute to the variability of the output of the neural network for samples not encountered during the training phase. Hence, theoretical studies cannot give accurate results with regard to such networks.

Thus, it can be concluded that the theoretical studies make several assumptions, that are not usually accurate in practice, to obtain results. Also, another main reason for their pessimistic prediction of bounds on the generalization error, is due to considering

simulated/synthetic random pattern associations which cannot be generalized without a large number of examples.

2.6 Summary and Conclusions

In this chapter we have briefly reviewed some of the recent theoretical results on the generalization capability of neural networks.

Since it is not possible to study theory on generalization without quantifying generalization, we initially gave an overview of some measures of generalization.

Then we described a model of learning from examples used in theoretical studies. Some attempts to provide a theoretical framework for the concept of generalization were discussed. To provide a theoretical framework for the study of generalization in pattern recognition tasks, synthetic models of pattern recognition tasks are assumed. The training data is generated from these models with some assumed characteristics. Measures of generalization are proposed based on some objective criteria, and the performance of generalization has been obtained in stochastic sense as closed form expressions as a function of parameters of the model as well as the training set. All these studies are made using theoretical formulations without actually training the pattern recognition system and testing it for its performance. Some results obtained from computational learning theory and behavior of learning curves were reviewed. The relation of generalization ability of a network to the capacity of the network and the size of the training set used to train the network is given.

Reasons for the limited applicability of the analytical results as tools while designing neural networks to solve pattern recognition problems are discussed. The difference between real world pattern recognition problems and pattern associations with synthetic data used in analytical studies is one of the main reasons for this drawback. Despite these limitations, the theoretical results give us some idea about the extent of influence of size and architecture of neural networks and size of training set on generalization.

Chapter 3

GENERALIZATION IN FEEDFORWARD NEURAL NETWORKS

3.1 Introduction

Many models of neural networks have come into existence over the past few decades [58], [33], [25]. Most of these models have evolved from basic models like the Hopfield network [19] and perceptron [18] which were adapted to solve specific problems. The neural network models can be classified into feedforward and feedback models. In this thesis we concentrate on feedforward models of neural networks that use supervised learning. Fig.3.1 illustrates a typical feedforward neural network with 3 inputs, 3 outputs and 4 hidden nodes. The weights of the network are adjusted using supervised learning.

Supervised learning is a method of learning in which the training set consists of input-output pairs. These input-output pairs may be any arbitrary association of some input with an output. These pairs of examples may correspond to a mapping function, or a set of patterns and the classes to which they belong. In supervised learning the error between the network output and the target output is reduced in order to obtain desired output for the corresponding input.

Formally, the training set of size k can be represented as $T_k = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_k, \mathbf{y}_k)\}$ where $\mathbf{x}_i \in \mathbb{R}^n$ are the input vectors of dimension n and $\mathbf{y}_i \in \mathbb{R}^m$ are the output vectors of dimension m , and \mathbb{R} represents the set of real numbers. Let $f_{\mathbf{w}}$ represent the function realized by a neural network with weights \mathbf{w} , then our aim in supervised learning is to adjust the weights such that, $f_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{y}_i$, $\forall (\mathbf{x}_i, \mathbf{y}_i) \in T_k$, under the assumption that the training examples are noise free. If the examples are not noise free, then the error between $f_{\mathbf{w}}(\mathbf{x}_i)$ and \mathbf{y}_i for all the training examples is minimized.

This kind of learning process is especially useful in the case of pattern association

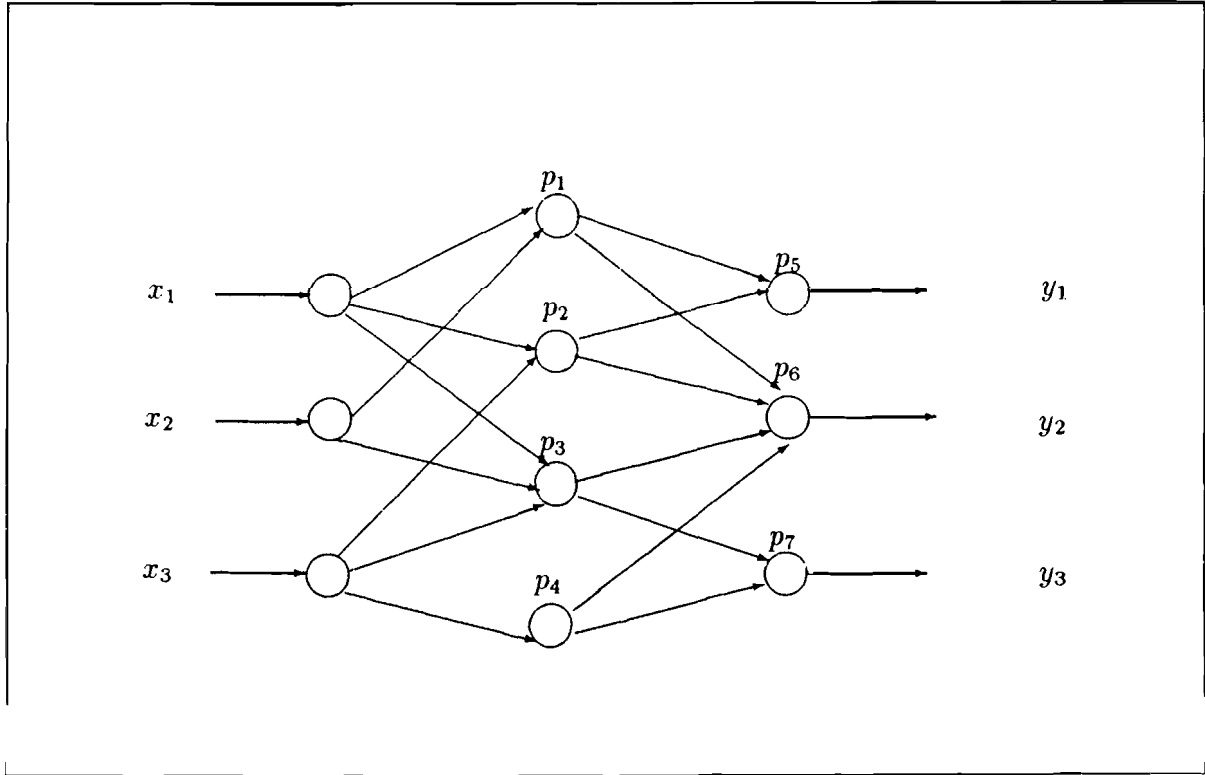


Figure 3.1: Typical feedforward neural network with 3 inputs x_1 to x_3 and 3 outputs y_1 to y_3 . The processors or nodes are denoted by p_1 to p_7 .

problems, where inputs and corresponding desired outputs are known.

From the above discussion we observe that objective function proposed for theoretical study has been applied to learning in neural networks from examples of data. In this case, only the training data is available and no model is assumed for the pattern recognition system. A criterion based on the objective function is used to fix the weights of the neural networks during the training phase. A cross validation measure is used for testing the generalization ability of the neural network from the given training data. The generalization performance is poor for training samples consisting of purely random data. Even when the training data belongs to the generalizable problem, the ability of a neural network to generalize depends critically on the nature of the problem, the number of parameters in the network, the number of training examples, the objective criterion used in training, the manner of presentation of examples in training, etc. Moreover, cross validation measure itself may not be adequate to evaluate the generalization behavior of the network. In this chapter we study **generalization** capability in feedforward neural networks. We concen-

trate on the property of generalization of the network without taking into consideration the problem that is being solved by the network.

In the section 3.2 we discuss the issues related to generalization capability of feedforward neural networks. Generalization can be improved using a problem-independent approach or by taking into consideration knowledge about the problem. We discuss the problem-independent approaches to improve generalization in section 3.3. Some problem-independent methods for improving generalization are presented in section 3.4. The limitations of problem-independent approaches to improve generalization are also discussed in this section. Issues in quantifying generalization are described in section 3.5 and a new measure of generalization which makes use of fuzzy theory is proposed. We give a summary of this chapter in section 3.6.

3.2 Feedforward Neural Networks - Limitations in the Context of Generalization

Pattern recognition tasks are usually complex, and cannot be solved by designing a single algorithm that takes care of all the variations in the patterns [32]. Therefore, methods of learning from examples have developed. Learning algorithms generally perform better in lower dimensional space. Thus, it is important that the patterns can be transformed to lower dimensional space so that the learning can be performed well. This enables us to view the pattern recognitions tasks as consisting of two parts, namely, a feature extraction part and a pattern association parts. Feature extraction is problem-dependent. The performance of the pattern associator depends on how well the features are chosen by the designer. The neural network approach to pattern recognition tasks tries to overcome this dependence on a designer for selection of features from the patterns. For this purpose a learning algorithm is used to adjust the weights of a feedforward neural network using training examples. It is hoped that the neural network is able to extract certain features by itself (without aid from a designer), from the training examples as a result of learning, and generalizes to give the desired output for new samples. But in most of the implementations of learning, the goal of the learning process is to minimize an objective function which has been obtained from analytical studies on synthetic data [55]. Thus, the method does not take into consideration extraction of features. This is one of the major limitation of

generalization in most neural **network** methods used for pattern association.

Despite the above limitation, neural networks perform reasonably well for **pattern** association problems because of their ability to learn complex mappings in higher dimensional space. This generalization performance of a neural network is improved by manipulating parameters of the network, which include:

- Architecture of a neural network
- Training set - size and quality
- Learning algorithm
- Criterion for stopping training

These approaches to improve generalization are discussed in detail in the following section.

3.3 Approaches to Improve Generalization

Even though ANNs have the limitation of not being specifically designed to capture features and to generalize, several efforts have been made to improve generalization performance of neural networks using an objective criterion for training data, and error rate on test data for measuring the generalization capability of the network. The generalization performance is evaluated by varying the free parameters of the neural network and the learning algorithms. In this section we discuss some approaches to improve generalization considering the key issues in generalization that they try to overcome.

The methods suggested to improve generalization in neural networks may be of two types:

- **Problem-Independent:** These methods deal with the functioning of a neural network, method of presentation of data, etc.
- **Problem-Dependent:** These methods include special design of a neural network taking into consideration available knowledge about the problem [14], [45].

In this chapter we concentrate on the problem-independent approaches to improve generalization. The problem-independent methods manipulate the parameters of the network as follows:

- **Architecture of Neural Networks:** Neural networks can be thought of as non-parametric estimators of functions from a given set of values of the functions, namely, the training set. Nonparametric estimators do not make assumptions of a model for the training data. To be truly nonparametric, we should use large networks that give more flexibility to the functions realizable by learning from examples. Many training examples are required to achieve good generalization when a large network is used. But it is not always possible to obtain a large number of examples to train the network. Hence, there is a need to reduce the number of parameters of the network such that the available training set is sufficient for good generalization. The architecture of the neural network controls the number of parameters. Hence, choice of an optimum architecture is one of the major approaches to improve generalization. One of the key existing methods of optimizing the architecture is pruning which is discussed in detail in the survey paper by Reed [47].
- **Training Set - Size and Quality:** As discussed above it is advantageous to have a large number of training examples to train the network. Methods, that manipulate the training set such that more data is available to train a network, are extensively studied for improving generalization. One such method is introduction of noise into the training examples to generate new training examples which can be used to train the network [24]. It has also been shown that the training set has to be a good representation of the examples that occur in the problem being addressed to ensure good generalization [36].
- **Learning Algorithm:** In many existing models of neural networks it is not possible to train the neural network when the number of examples is large. The time required to train the network becomes very large. So, methods to accelerate the learning algorithm are studied in an effort to train the network with large number of examples in finite time [29].
- **Criterion for Stopping Training:** Decrease in error on the training set during learning phase does not always ensure good performance on a test set. The phenomenon by which decrease in training error results in increase in generalization error is termed as *overtraining*. Fig.3.2 gives generic graphs that show the behavior of training and test error with number of training iterations. Overtraining occurs be-

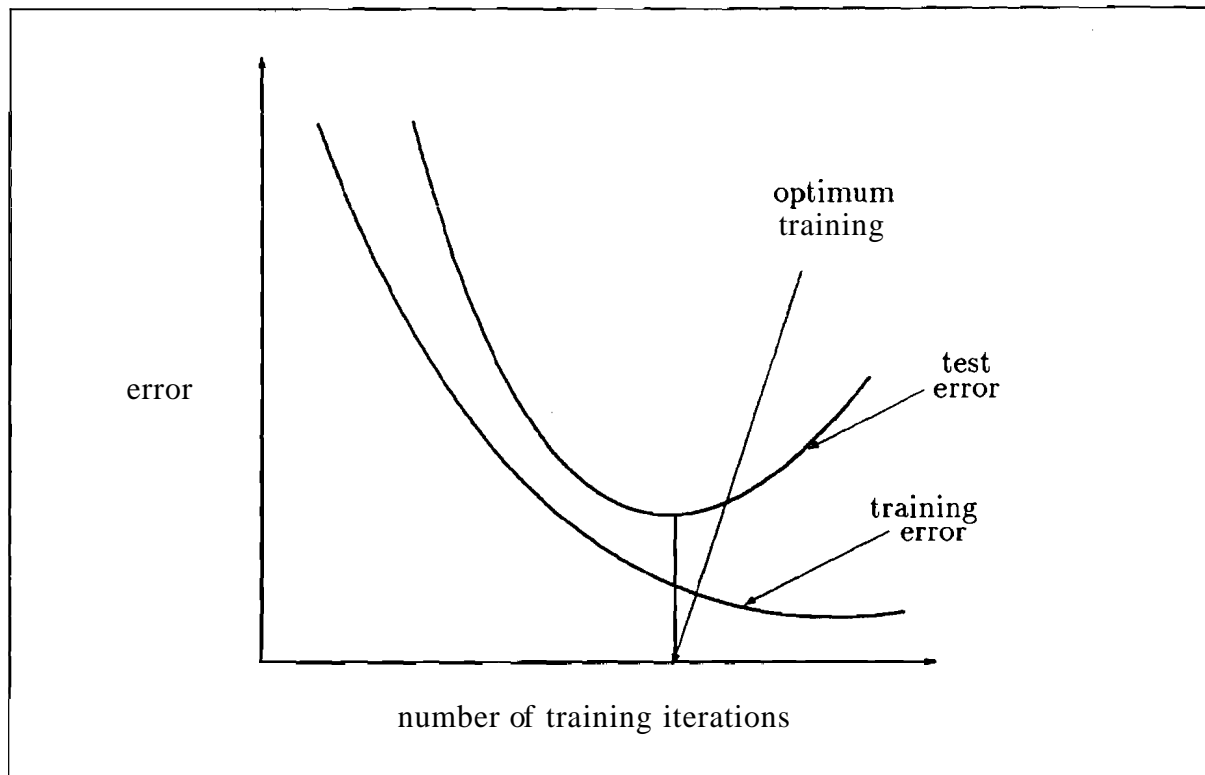


Figure 3.2: Graph depicting *overtraining*. There is an increase in generalization/test error even though the error on training set decreases with larger number of training iterations.

yond the optimum point indicated. Overtraining is attributed to the memorization of training examples by the neural network due to overfitting of function realized by the network to the noisy training data examples. Finding of a criterion for stopping training when generalization is the best, and thus avoiding overtraining, is a key issue of the generalization in feedforward neural networks.

In the following section we propose some methods to improve generalization which are independent of the problem to which the network is being applied.

3.4 Suggested Methods for Improving Generalization

As already discussed in the previous section, we can improve generalization by manipulating certain parameters of the network. In this section we study the effects of different stopping criterion and also the manner of presentation of data on the performance of

generalization. We show that the proposed stopping criterion and methods of presentation of training data may sometimes improve the performance. But in such cases the improvement is marginal.

In subsection 3.4.1 we propose a new stopping criterion for training. In subsection 3.4.2 we give a method of using multiple blocks of data for weight updation during the training phase. The use of two training sets, one to train the network and another to perturb the weights once the network is trained, improves generalization capability of the neural network. This method is discussed in subsection 3.4.3.

In this section we consider the feedforward neural network trained using backpropagation algorithm to study the effectiveness of the approaches to improve generalization in the context of a classification problem. The problem of classification of vowels 'a', 'e', 'i', 'o' and 'u' uttered by three different speakers is addressed. We use formants which are resonances of the vocal tract as features. The input is a three dimensional vector consisting of the first three formants. We train the neural network on a training set consisting of three hundred examples. The generalization capability of the neural network is evaluated based on its performance on a test set which consists of examples that did not occur in the training set. In the experiments conducted in this section we used a test set consisting of 1800 examples.

3.4.1 Stopping Criterion for Training

The method of updating weights does not ensure that there is an improvement in generalization. That is why overtraining occurs. In literature we find various stopping criteria. used to stop the learning process [25]. Some of the criteria used are:

1. *Magnitude of gradient*: This method is used in gradient descent approaches of learning. Here the learning algorithm is terminated when the magnitude of gradient is sufficiently small, since by definition the gradient is zero at the minima.
2. *Cost function below certain minimum value*: When the cost function is minimized during the training process, a certain threshold value is chosen, and training is stopped when the cost function value is below this value. However this requires the knowledge about the minimal value that the cost function can reach, which is not usually known. In pattern recognition problems one can stop training as soon as all

the training data are classified correctly. Many times the network may not manage to classify all the examples in the training set correctly, and even if it does, this does not ensure that it will give its best performance on the test set.

3. *Fixed number of iterations:* Here the training is stopped after a fixed number of iterations. This method does not guarantee that the algorithm terminates when the best solution is reached.
4. *Performance on test set:* Here the data is split into two sets: a training set which is used to train the network and a test set which is used to measure the generalization performance of the network. During learning the performance of the network on the training set continuously improves but its performance on test set improves to a certain point, beyond this point it starts degrading. At this stage the network begins to overfit the training data, and so training is stopped. This method is sometimes called cross validation, but it should not be confused with the actual cross validation which has already been discussed in section 2.2.2.

The first three criteria are sensitive to the choice of parameters, and if not chosen properly the results can be very poor due to premature termination of training. The fourth method, however, does not suffer from this kind of premature termination, but results in good generalization performance of the network. However, checking performance on a test set is computationally intensive. Further, if the number of data samples is limited, it reduces size of the training set.

In this section we propose an alternative stopping criterion for feedforward networks used for pattern classification. It can be shown that when the least square error is minimized during training, the outputs of the network tend to converge to the *a posteriori* class probability $\pi(c|\mathbf{x})$ [39]. The proof of this result is given in Appendix A. It is also observed that although there is no explicit constrain imposed on the sum of the outputs of the neural network, its value tends to become close to unity when error between network output and desired output is small. In the proposed stopping criterion we make use of this observation and stop training when the sum of the outputs of the network is within a 'small' closed interval $[a_1, a_2]$ around 1, instead of the error between network output and desired output tending towards zero. An example of a typical values of the interval $[a_1, a_2]$ is $[0.95, 1.05]$.

The use of this method does not force the network function to fit all the data points exactly. Therefore it helps to alleviate the problem of overfitting. The performance of a network that uses this stopping criterion is given in Table 3.1. It can be observed that it does give only a marginal improvement in the performance on the test set. Especially when the network is small, there is no improvement as there is no scope for overtraining. But in the case of large networks there is comparatively more improvement in generalization. This method may be suitable when there are a large number of outliers in the training set., which if learned till the network gives low error, cause the generalization to be poor.

Sl. No.	Number of Hidden Nodes	Percentage Correct Classification	
		Error Reduction Stopping Criterion	Suggested Stopping Criterion
1	5	86.3%	86.3%
2	10	87.2%	81.2%
3	40	88.8%	90.7%
4	50	89.4%	90.1%
5	60	89.4%	91.0%
6	70	89.2%	90.8%
7	80	88.7%	91.2%
8	90	88.2%	92.8%

Table 3.1: Comparison of error reduction stopping criterion with suggested stopping criterion.

3.4.2 Variable Block Size Update Mode

There are two modes of updating weights in the feedforward networks that use gradient descent algorithms for learning, namely, the pattern mode and the block mode [18]. In the following discussion we restrict our attention to the backpropagation learning algorithm. In the pattern mode the weights are updated after the presentation of each

pattern, whereas in the block mode the weight changes are accumulated till the end of one cycle through all the training examples and then the weights are updated. These two methods have been studied extensively, and each of these methods has its advantages and disadvantages [19]. The **pattern** mode requires less memory and, since the patterns are presented in random to the network, the search for the **solution** is stochastic and there is less chance of the network getting stuck at a local minima. On the other hand, in the block update mode the estimate of the gradient vector is better; so each updation of weights generally results in decrease in error without much oscillation.

In the proposed method we combine the advantages of both these methods by using blocks of data to update the weights. In the normal block update method the whole training set is treated as a single block and, updating of weights is done after the whole training set is presented to the network and weight changes are accumulated. In the proposed method we treat the training set as consisting of blocks, and the weight updation is done at the end of each block by considering all the changes in weights for that block. The behavior of neural networks, after training them with varying size of blocks of input data, is investigated here. In this approach, at each stage a better estimate of the gradient is made and also the updation retains its random nature because the blocks of data are still random as they consist of a few **patterns**.

It has been reported that when presentation of the data is in random from the data set, there is an improvement in the generalization performance. Here we investigate use of the same patterns in each block for repeated training iterations and also the use of different patterns in each block for repeated iterations. We call the former method as **fixed block** mode and the latter as **random block** mode.

Table 3.2 gives the comparative performance of the fixed block and random block modes of weight update. Column one gives the performance of fixed block mode and column two gives the performance of random block mode for various sizes of the blocks of training data set consisting of **300** examples. When pattern by pattern mode of update is used, the network classifies 87.7% of the test set samples. From the table it can be inferred that when blocks of data are used to train the network, the performance on the test set improves. This can be seen from the first six rows of the table. But as the size of blocks increase the generalization, as evaluated by error rate on test set, decreases. Also, it is observed that use of the random blocks of data to update weights results in marginally

Sl. No.	Size of Block	Percentage Correct Classification	
		Fixed Block Mode	Random Block Mode
1	5	90.0%	91.1%
2	10	91.0%	93.1%
3	20	91.1%	93.8%
4	30	91.4%	93.2%
5	40	90.3%	93.5%
6	50	88.1%	93.7%
7	100	78.0%	77.6%
8	150	74.6%	75.0%
9	200	70.5%	70.5%
10	250	66.8%	64.4%
11	300	69.7%	69.7%

Table 3.2: Comparison between the fixed block and random block modes of update of weights. Training was done for 5000 iterations on a training set consisting of 300 examples. The performance for pattern by pattern update is 87.7%. The test set consisted of 1800 examples.

better generalization than the fixed blocks of data. However, it should be noted that, as the size of blocks increases the performance of the random block mode decreases compared to that of fixed block mode for the same number of training iterations.

3.4.3 Weight Perturbation with Training Examples

One of the main issues of generalization in feedforward neural networks is overtraining. In this situation the network constructs a curve that passes through most of the data samples, and this results in poor generalization because the data is usually noisy. Fig.3.3 illustrates poor generalization due to overfitting. This problem of overfitting, that occurs when error on training set reduces, has motivated the suggestion of using two sets of

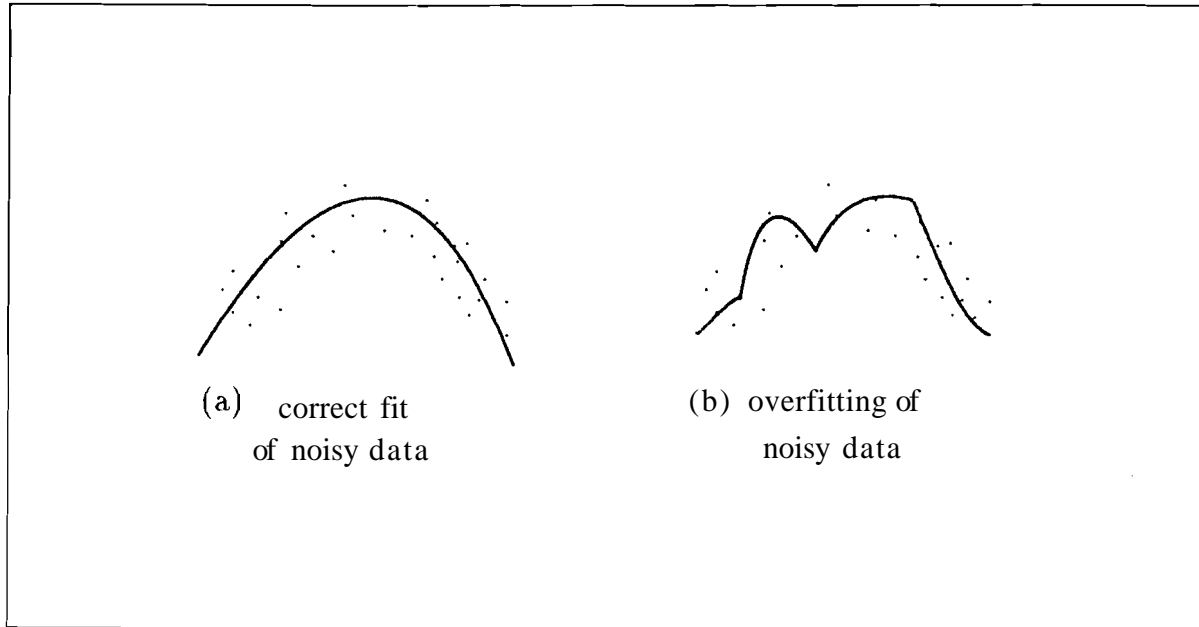


Figure 3.3: Illustration of *overfitting*. (a) Actual function that fits the noisy data. (b) The curve that fits the data well, but is a poor approximation of the actual function.

examples to train the network. In this method the network is trained on one set of examples till low error is reached, and the second set of examples is used to train the network for a few number of iterations. The reason for performing a few iterations on the second set of training examples is to perturb the weights that have been obtained by training on the first training set. This perturbation will disturb the function learned by the network which may be overfitting for the first training set. Since the perturbation is done according to the examples of the function being approximated, it is hoped that the resulting disturbed function approximates the actual function in a better way.

Table 3.3 illustrates the performance of the network for which this method of presentation of data is implemented. The first row of the table gives the performance of the network when it is trained on a single training sets. The successive rows indicate the performance for various sizes of the first and second training set. It has been observed that the performance is usually better when the second training set is small. Moreover, the performance usually decreases with increased number of training iterations on the second training set. In the table the performance value has been obtained by performing about 20 training iterations on the second training set. It should be observed from the

Sl. No.	Size of first training set	Size of second training set	performance of network
1	300	0	87.7%
2	250	50	92.6%
3	200	50	91.4%
4	200	100	87.1%
5	100	20	90.7%
6	100	50	91.8%
7	100	100	77.4%

Table 3.3: Performance of neural network trained with two training sets. The first row of the table indicates the performance when a single training set is used. Successive rows indicate the performance for different sizes of the first and second training set.

third, fourth, fifth and sixth rows of the table that a better performance is achieved by the network trained by the **proposed** method even though a small training set is used. For example, we achieve 91.8% classification by using 150 examples as indicated by the sixth row 'entry of the table as against the 87.7% classification achieved by using 300 examples as a single training set.

3.5 Quantification of Generalization

In section 2.2 some limitations of measuring generalization were stated and a brief overview of some measures of generalization was given. In this section we look at some of the deficiencies in the existing approaches to evaluate the performance of generalization. We discuss an intuitively appealing approach to judge the generalization behavior of a network, although the proposed approach has limitations to apply in practice. The approach is based on using the concepts of fuzzy sets and also the newness of test samples. A brief overview of relevant fuzzy set concepts is given in Appendix B.

3.5.1 Issues in Measure of Generalization

The difficulties that arise during the measure of generalization are:

- **Lack of Proper Definition of Generalization:** We come across two definitions of generalization [25] that are used in related studies, namely:
 1. Ability to produce accurate results on *new examples* not present in the training set.
 2. How well the network performs on the *actual problem* once the training is complete.

There is a subtle difference between these two definitions. This difference can be highlighted by considering the case when generalization is considered to reach a high value in by each of these definitions. In the case of generalization given by the first definition, the value is high when the network gives desired output for samples that did not occur in the training set. Here it is independent of probability of occurrence of the samples. In contrast to this, generalization given by the second definition is high if the network is able to yield the desired output for the most frequent examples that occur in the problem being addressed. It indicates the ability of the network to give the desired output for any input from the input domain. This quality is influenced by probability of occurrence of the sample.

- **Need to Consider Quality of Training Set:** Generalization in neural networks depends on how well the training set represents the input domain. Further the output error is minimized during the training phase of the network so that the network gives the desired output for the training examples. Consequently, if the test set contains samples similar to the training examples, then the network is able to classify them properly. Thus, the generalization capability of the network appears to be high if we measure generalization using methods which do not take into consideration quality of the training set. Therefore, there is a need to consider the quality of training set used to train the network, and accordingly interpret the performance on test set,
- **Inherent Fuzzy Nature of Generalization Measure:** When a trained neural network is tested on new samples, its output may not be exactly what is desired

or completely different from the required output, instead it may be correct to some extent, i.e., partially correct. All existing approaches to measure of generalization capability do not make allowance for the partial correctness, they only consider whether a sample gives the desired output or not. The suggested measure takes partial correctness into consideration by the use of fuzzy approach.

In the following section we propose a method of quantifying generalization that tries to take into account these limitations of measuring generalization.

3.5.2 Fuzzy Generalization Index

The generalization index developed in this section quantifies the ability of a neural network to produce the desired results on test samples not used in the training set. The formulation of this index takes into consideration both the fuzzy nature of generalization and the influence of the size and quality of the training set. The objective of measuring generalization capability of a network is to determine the extent to which we can rely on the output of the network for all the samples in the input domain of a given problem, once the training is complete. Since the number of patterns of the input domain is usually large, we try to evaluate generalization based on a finite set of test samples. In order to accomplish this task, we compute the network output for each test sample and compare it with the desired output.

Based on the information obtained from each test sample, an estimate is made about the generalization capability of the network. Fusion of evidence obtained from each test sample is achieved using a fuzzy aggregation operator. To derive an index for generalization (\mathcal{G}), there is a need to calculate the extent of generalization (g_i) for each test input (\mathbf{x}_i).

Before presenting the proposed method of measuring generalization, some elementary properties that should be satisfied are stated. The following notations are used for stating the properties:

Z = set of all possible input-output pairs

T_k = $\{(\mathbf{x}_i, \mathbf{y}_i), i = 1, 2, \dots, k\}$ is the training set, $T_k \subseteq Z$

S_n = set of n test input-output pairs, $S_n \subseteq Z$

g_i = generalization value attributed to the i th test sample pair $(\mathbf{x}_i, \mathbf{y}_i)$

\mathcal{G} = generalization index

1. $\mathcal{G}=0$ if and only if no test samples other than the examples in the training set give the desired output. That is,

$$g_i = 0, \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \notin \mathbf{T}_k \Leftrightarrow \mathcal{G} = 0 \quad (3.1)$$

2. $\mathcal{G} = 1$ if and only if $g_i=1$ for every new test sample. Hence,

$$g_i = 1, \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \notin \mathbf{T}_k \Leftrightarrow \mathcal{G} = 1 \quad (3.2)$$

3. There are no constraints on accuracy of learning on the training set imposed by \mathcal{G} .
4. \mathcal{G} is not influenced by the probability of occurrence of the examples.

The first property substantiates the fact that the generalization index quantifies the ability of the neural network to classify new samples. Therefore, it states that when no new sample gives the desired result, the index is zero. An interesting implication of this property is that $\mathcal{G} = 0$ when $\mathbf{T}_k = \mathbf{Z}$, because there is no test sample $(\mathbf{x}_i, \mathbf{y}_i) \notin \mathbf{T}_k$. This is intuitively appealing, because, when all possible examples are used in training the network, there is no need for generalization. The second property states that every new example should give a value 1 for the generalization index. The third property highlights the fact that the performance on training set can not significantly affect the index \mathcal{G} . The final property requires that \mathcal{G} is independent of the frequency of occurrence of samples.

The proposed method of measuring generalization is now presented. First, the value g_i for each test sample is determined. The generalization value g_i attributed to the i th test sample depends on two factors, namely, "*how new the test sample is*" and the "*degree of correctness*". Degree of correctness makes allowance for the fact that the output may be partially correct. The second factor enables us to account for the quality of the training set. The factor "*how new the test sample is*" also indirectly accounts for the size of the training set. It enables distinguishing between a network that requires large number of training examples to perform well and a network that requires a few training examples for similar performance. The concept of newness of a test, sample is fuzzy in nature, because

there is a gradual change from similar pattern to "more or less new" type of pattern. Consequently, fuzzy membership can be used to model the newness concept. We define newness of a pattern by μ_{new} , where $\mu_{new} : \mathfrak{R} \rightarrow [0, 1]$. Here newness of an example is represented by considering the Euclidian distance in the input space from the nearest training example belonging to the same class. This distance is normalized so that its value is always in $[0, 1]$. The membership value μ_{new} is transformed to $\mu_{mol\ new}$ (*mol* new represents more or less new) using a *fuzzy* hedge [40] operator to accommodate the fact that certain patterns are "more or less" new. The fuzzy hedge is modeled using the function [40],

$$\mu_{mol\ new} = (\mu_{new})^{\frac{1}{2^\beta}} \quad (3.3)$$

where $\beta \in \mathbb{N}$, here \mathbb{N} is the set of natural numbers.

Similarly, we can express the output of a test sample as a fuzzy membership. Let κ represents the normalized output error of the neural network. Then $(1 - \kappa)$ represents the correctness of the examples. This membership function can be transformed, if necessary, using a fuzzy hedge operator, to a membership function ($\mu_{mol\ correct}$). We observe that the generalization value of the network for a test sample should be high when both the degree of correctness and the newness of the test sample are high. In all other cases, generalization value for an example should be low. A fuzzy AND operator is used to combine $\mu_{mol\ correct}$ and $\mu_{mol\ new}$. This ANDing is realized by a multiplication operation. From the above discussion, it is clear that g_i is in $[0, 1]$. Hence g_i can be viewed as a fuzzy membership function with domain \mathbf{S}_n [10].

To determine the generalization of the network (\mathcal{G}), we combine the information obtained from g_i s of each test sample. We accomplish this task by using a fuzzy operator h_α [30]. Thus, \mathcal{G} is given by,

$$\mathcal{G} = h_\alpha(g_1, g_2, \dots, g_n) = \left(\frac{g_1^\alpha + g_2^\alpha + \dots + g_n^\alpha}{n} \right)^{1/\alpha} \quad (3.4)$$

where $\alpha \in \mathfrak{R}$ ($\alpha \neq 0$). The parameter α can be used to control the softness of the operator.

It is important to note that h is not a T-norm [9], [12], because, it does not satisfy a property of the T-norm, namely, $h(a, 1) = a$. In our case, it is essential that $h(a, 1) > a$ when $a \neq 1$ because 1 indicates perfect generalization of a particular test sample and

by combining this information with a, the generalization value obtained from another test sample, we should be able to infer that the generalization of the network is better than that inferred from a single test sample with generalization value a.

In the remaining part of this section we give a measure of generalization that quantifies how well the network performs on the actual problem once the training is complete. In this case, the distinction between training set and test set vanishes once the training is over.

We represent this generalization measure by \mathbf{M} and input space by S . The following are the fundamental properties of the measure:

1. It is minimum, i.e., zero, when no example is approximated correctly. Thus,

$$f(\mathbf{x}_i) \neq \mathbf{y}_i \quad \forall \mathbf{x}_i \in S \quad \Rightarrow \quad \mathbf{M} = 0 \quad (3.5)$$

2. The measure \mathbf{M} is 1 when all examples are approximated correctly. Thus,

$$f(\mathbf{x}_i) = \mathbf{y}_i \quad \forall \mathbf{x}_i \in S \quad \Rightarrow \quad \mathbf{M} = 1 \quad (3.6)$$

3. Generalization measure is influenced by accuracy of training on the training set.

Most measures of generalization that occurs in literature satisfy these properties. One such measure that we find in literature is defined as follows,

If input sample \mathbf{x} occurs with probability $\pi(\mathbf{x})$ in the input domain, $f_{\mathbf{w}}$ represents the function approximated by the trained neural network and $\mathbf{Z} = \{(\mathbf{x}, \mathbf{y})\}$ is the input domain, it follows that,

$$\mathcal{M} = \int_{\mathbf{x}} Pr[f_{\mathbf{w}}(\mathbf{x}) = \mathbf{y}] \pi(\mathbf{x}) d\mathbf{x} \quad (3.7)$$

The major difference between generalization index and generalization measure is that in the former case the error in output space is weighted by the "newness of sample" whereas in the latter case the error in output space is weighted by the "*probability of occurrence*" of the sample.

3.5.3 Results and Discussion

It is observed that the error rate does not consider the quality of test samples, therefore unless a large test set is used, there is a higher influence of bias of the test set on this measure. For example, if the test set consists of only test samples similar to the examples in the training set, the error rate measure gives a large value for generalization, and it gives a low value if the test samples are very different from the training examples, and thus do not result in the desired output for the network. There is no method that takes into account the bias induced by the type of test set being used to test the network. Unlike this, the generalization measure \mathcal{G} takes into account the possible bias in the test set by weighting the contribution of each test sample in it by a measure that depends on how different it is from the training examples. This also makes the measure more realistic. However, lower absolute value of generalization index may result. The lower absolute value of generalization is because the scope to generalize is lowered when the network is trained on many examples. This enables distinguishing a network, that is able to classify correctly after being trained on a few examples, from a network which gives the same performance after being trained on many examples.

Apart from this, when there are training examples in the test set used for the measure of generalization, the g_i s for these examples are zero, because newness of the test samples is zero. As a result, when there are a large number of training examples in the test set the value of \mathcal{G} calculated by aggregating the g_i s decreases, and in the case when the number of training examples in test set reduces the value of \mathcal{G} increases. This is because the test set with a large number of training examples is biased towards giving high generalization value for the network, though no training example can be used to evaluate the generalization capability of the network because the nature of training ensures good performance on training examples. This is unlike the error rate calculated on the test set, where if training set is used as test set then it results in high value of generalization capability. But we know that we can not conclude anything about the generalization capability of network by checking its performance on training set only. Although, it is not always correct that the generalization value is low when test set consists of training examples, we feel that it may be a better alternative to consider it low than to be misled into the belief that the network generalizes well after testing with a set of examples biased towards the training set.

Experimental observations from an implementation of the generalization measure developed in this section, for the problem of Opening Bid in Contract Bridge game are given in Appendix C.

In the following subsection we give some limitations of the proposed generalization index.

3.5.4 Limitations of Generalization Index

Although the proposed generalization measure apparently takes care of the limitations of generalization measures discussed in subsection 3.5.1, its application to real world problems is very limited because it assumes that newness of examples can be measured. In our implementation of the measure we have used Euclidian distance in input space as a measure of newness. As a result, this measure of newness is applicable only in cases when similarity of patterns is reflected as closeness of examples in terms of Euclidian distance in input space. This limits the applicability of the proposed measure, and as a result, we do not apply it into the pattern recognition problems discussed in the later chapters.

3.6 Summary

In this chapter we presented the current view of generalization in feedforward neural networks. We have identified the limitations of generalization in feedforward neural networks and suggested some methods for improving generalization. We have mentioned briefly the disadvantage of not considering the problem specific knowledge to improve generalization.

A measure for generalization is proposed which takes into account the fuzzy nature of generalization and quality of training set used to train the network. The application of the measure is limited to the tasks where the closeness of samples in space reflects their similarity. Hence, it has limited application in real world pattern recognition tasks.

Chapter 4

GENERALIZATION AS A PROBLEM DEPENDENT PHENOMENON

4.1 Introduction

Methods to improve generalization can be viewed as either problem-independent or problem-dependent. Problem-independent methods make use of a general structure of neural networks and manipulate the parameters of the network to obtain improved generalization. Such methods have been discussed in chapter 3.

On the other hand, neural networks that use knowledge about the problem can be developed to achieve good generalization performance. In this case, knowledge about the problem is incorporated into the network in the form of constraints on the parameters of the network. Examples of such networks developed in a problem specific way are found in [45], [14], [59].

In this chapter we focus on problem-dependent approaches to improve generalization. Section 4.2 discusses the problem-dependent nature of the generalization phenomenon. In section 4.3, we describe the desirable generalization behavior due to incorporation of knowledge into the neural network by analogy with modeling a system represented by data. In section 4.4, we discuss how knowledge is incorporated into the network by considering Radial Basis Function Neural Networks (RBFNN). Experimental observations regarding the comparative performance of RBFNN and MLP in the context of classification and function approximation are given in section 4.5. We summarize the chapter and state the conclusions in section 4.6.

4.2 Generalization in the Context of Specific Problems

Problem-independent methods of improvement of generalization do not take into consideration the problem specific knowledge that may improve generalization significantly. For example, one of the methods by which generalization is improved in a problem-independent way, is by manipulating the data by adding noise, so that there is apparently more data available for the training process. But it is not always possible or meaningful, to obtain more data for training by adding noise. This can be illustrated by considering a pattern mapping/function approximation problem where a few scattered training examples are given. In this case it is not possible to obtain a better approximation of the function by using more training examples created by adding noise. This is because, it does not give us insight into the behavior of the function at points where training examples are not given. But if we have some additional knowledge about the function behavior, then we can get a good approximation of the function by incorporating this knowledge into the network. This is illustrated in Fig.4.1.

Thus, we note that many approaches to improve generalization do not correspond to the human concept of generalization. Most methods dealing with data attempt to load the data into the neural network, rather than capturing the pattern behavior in the input data, leading to a tendency to memorize the input data by the neural network. This drawback can be overcome by considering problem specific knowledge. Application of problem knowledge along with data should bring out a trained system whose behavior is analogous to other problems involving stochastic or deterministic modeling.

The following section substantiates the problem-dependent approach with an analogy from modeling a system represented by data.

4.3 Analogy with Modeling a System Represented by Data

Modeling a system, based on data obtained from it, is one of the most common problems encountered in engineering control systems [52] and pattern recognition. In many real

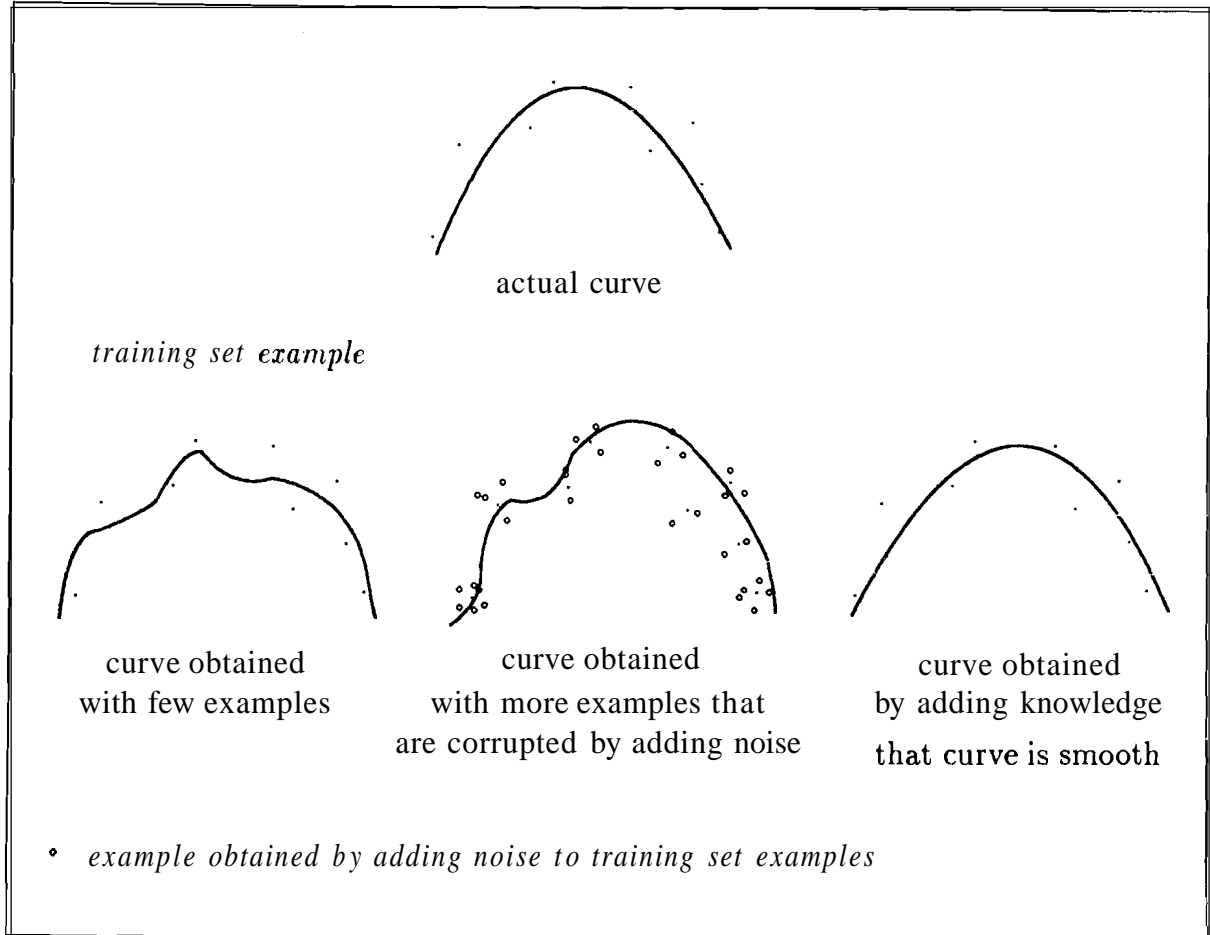


Figure 4.1: Illustration of better generalization obtained by incorporating knowledge about the problem.

world situations we have access only to the data generated by the system and, attempts are made to model the system from the data and analyze the system characteristics.

In the case of modeling, a model is first assumed based on some knowledge about a system that is being modeled. Then the data obtained from the system is used to determine the parameters of the model. It is observed that the modeling of the system is good when the model selected is a good representation of the system, and a large number of examples, obtained from the system, are used to evaluate the parameters of the model. Such a system is usually better than a model whose parameters are evaluated by using less data obtained from the system.

We take the examples of Linear Prediction (LP) analysis [37] and Hidden Markov Model (HMM) [44] to discuss the above observation.

Linear Prediction (LP) analysis is one of the aspects of time series (signal) analysis. The goal of this analysis is to model the system that generated the signal. The model that is developed can be used for prediction or forecasting, control, etc. In this method of modeling, the signal s_n is considered as the output of some unknown system with some input u_n such that the following relation holds [37]:

$$s_n = \sum_{k=1}^p a_k s_{n-k} + G u_n \quad (4.1)$$

where a_k , $1 \leq k \leq p$ and the gain G are the parameters of the hypothesized system. The above equation implies that the signal s_n is predictable from a linear combination of past p outputs s_{n-1} to s_{n-p} and input u_n . The order of the model is said to be p , where p is the number of previous signal samples that are used to predict the present signal. Various methods of estimation of the parameters, a_k , for $1 \leq k \leq p$ exist and are given in the tutorial review by Makhoul [37]. Here we briefly discuss the influence of the order of model chosen on the modeling of the signal.

It is observed that the prediction of signal is good irrespective of the order of prediction (p), as long as $p > p_m$, where p_m is the order of linear prediction for that particular signal. In the case of linear prediction, the modeling of the signal usually improves with increase in n , where n is the number of samples of the signal used for minimization of error energy for estimation of LP coefficients. The equivalents of p in neural networks are the number of weights and of p_m is the number of weights that are sufficient to solve the problem. The equivalence of n in neural networks is the number of examples used to train the network. Therefore, once the network is of sufficient size and the input data set is a good representation of the input domain, it is desirable that a network converges to the solution with high generalization capability.

In the LP analysis better estimation of the model parameters can be viewed as better generalization, as it results in the model giving a response closer to the actual system in all cases, including the situations where the actual system behavior was not given in the form of data used to estimate the parameters of model. Thus, an analogy from modeling of system represented by data is used to define a generalization behavior that is desirable in the case of neural networks.

Modeling of signals can be broadly categorized into deterministic and statistical models. The LP model discussed above is a deterministic model. In the following discussion

we consider the Hidden Markov Model (HMM) which is a statistical model. The underlying assumption of statistical model is that a signal can be characterized by a parametric random process of which the parameters can be estimated in a precise, well-defined manner unlike the deterministic models where some specific property of the signal is exploited. A HMM is a doubly stochastic process with an underlying stochastic process that is not observable, but can only be observed through another set of stochastic processes that generates the sequence of observed symbols [42].

An HMM is characterized by [44]:

1. A finite number of states.
2. A finite number of distinct observation symbols per state.
3. A transition probability distribution.
4. Observation symbol probability distribution.
5. An initial state probability distribution.

The parameters of the HMM are estimated by making use of examples generated by the system that it is modeling. Once the above characteristics of the HMM are estimated, it can be used as a generator to give a sequence of symbols. Thus, it is able to model the behavior of the system that generated the examples. It is observed that the modeling of the system is better when a larger number of examples are available to estimate the parameters of the HMM. This kind of behavior is desirable in the case of neural networks. Thus in modeling, it is observed that as the number of examples available to evaluate parameters of the system increase, the model behavior approaches closer and closer to the actual system. In contrast to the above, we find that this is generally not true in the case of neural networks. The main reasons for this are:

- When the network is smaller than the required size, it is found that the learning algorithm should ensure that the neural network arrives at a solution that gives the best possible generalization given its limitations due to its size. The problem with present neural networks learning procedures is that the network does not converge to a solution when it is not of sufficient size, and hence, there is apparently no solution when the network is smaller than the required size.

- When the network is larger than the required size, it should be possible to ensure that all the weights are involved in determining the desired output. The key issues in this case include:

1. Lack of sufficient training data leads to:

- *Overfitting*: the problem of overfitting has already been discussed in sections 3.3 and 3.4, and therefore, it is not elaborated here.
- *Unconstrained Weights*: When the training set is small, all the weights are not involved in the training process resulting in high variance in output of the network. This happens usually when very large networks are used because of the weights do not contribute to the output of the network. The weights balance each others effect on the output for the training set. This has been mentioned in section 2.5. This results because most learning algorithms do not ensure that all the weights take part in the approximation of the function. Such free weights result in the variability of the function for test examples [51].

2. *Bias Vs Variance Dilemma*:. When a neural network is large, it can realize many kinds of functions to fit a given set of data. Hence it becomes necessary to include some knowledge about problem being solved to limit the number of functions to the more feasible ones with respect to the problem. This is done by incorporating constraints into the neural network. But incorporating constraints into the neural network limits the function realization capability of the network. Sometimes incorporation of constraints may be to an extent that it introduces a bias in the output of the network for all training examples. Thus, there is a trade off between bias and variance when the available training set is limited. In the ideal case both bias and variance of the network can be made low when large amount of data is available to train it.

In this chapter we do not provide solutions to these issues but we discuss some existing models in the context of the ideas presented here. In the following section we discuss Radial Basis Function Neural Networks (RBFNN) highlighting the advantages that result from problem-dependent design of the network.

4.4 Neural Networks with Problem-Specific Knowledge

In this section we discuss some neural networks that have been developed taking into consideration problem-dependent knowledge. First, we present the RBFNN that are designed specifically for classification tasks. We then consider the RBFNN using regularization which is used for pattern mapping tasks [41].

4.4.1 Radial Basis Function Neural Networks for Pattern Classification

An RBFNN is a two layer network consisting of a layer of hidden nodes and an output layer. Fig.4.2 gives an example of an RBFNN. The hidden nodes use radial basis-functions to compute the input to the node and a Gaussian is used to evaluate the output. The calculation performed at the hidden node is given by the following equation:

$$h_j = \phi\left(\left(\sum_{i=1}^n (x_i - \mu_{ji})^2\right)^{\frac{1}{2}}\right) \quad (4.2)$$

where h_j represents the output of hidden node j , ϕ represents a Gaussian, x_i $1 \leq i \leq n$ are the components of an input vector $\mathbf{x} \in \mathbb{R}^n$ and μ_{ji} is the i th component in the weight vector of the j th hidden node. The calculation performed by the output layer node is given by:

$$y_j = \sum_{i=1}^H \lambda_{ji} h_i \quad (4.3)$$

where H is the number of hidden nodes and λ_{ji} is the weights from the i th hidden node to the j th output node.

The RBFNN performs the required classification by forming clusters of the input data [11]. The structure exhibited by the data is exploited for clustering at the hidden nodes, and the class labeling task is done by the output layer. This method of classification is not like classification method in MLP, which performs the classification by constructing class boundaries using separating hyperplanes. Thus, in the case of RBFNN the emphasis is on regions of input space where data exist while in MLP the emphasis is on regions of input space where data does not exist so that class boundaries may be placed there. This

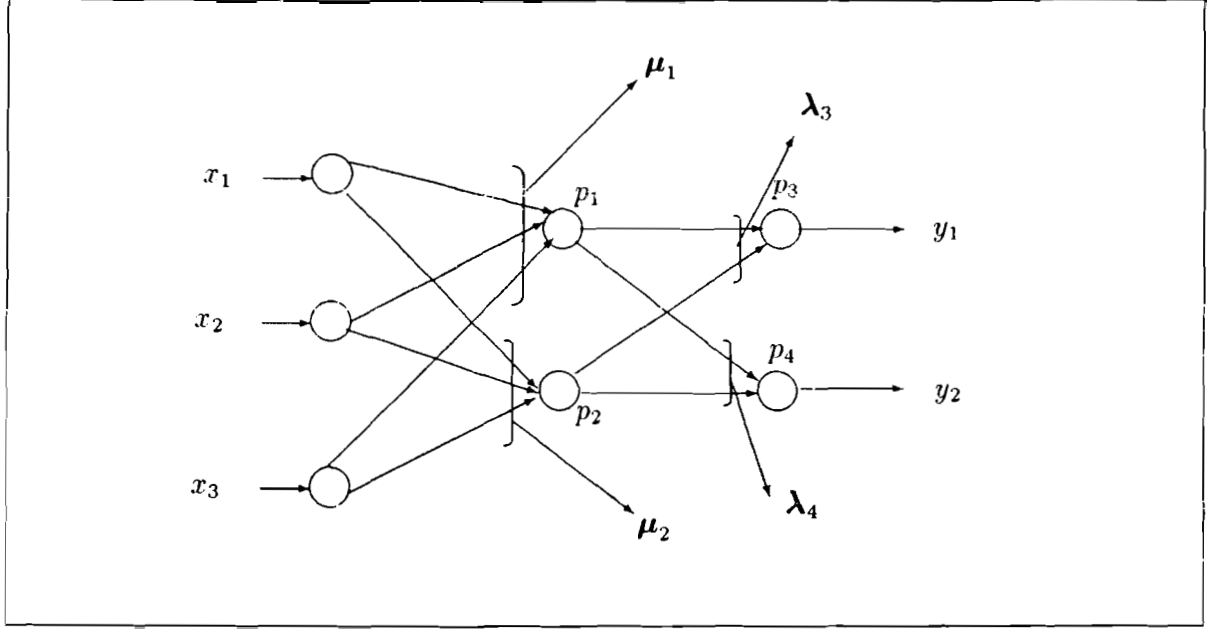


Figure 4.2: Radial Basis Function Neural Network (RBFNN). The nodes p_1 and p_2 are the hidden nodes which use the radial basis function, $(\sum_i (\mathbf{x}_i - \boldsymbol{\mu}_{ji})^2)^{\frac{1}{2}}$, to evaluate the input to the node. A Gaussian function is used in these nodes to obtain the outputs. The usual scalar product is used by the output nodes p_3 and p_4 .

distinction of processing at the first hidden layer nodes of RBFNN and MLP are given in Fig.4.3 and 4.4.

Fig.4.3 illustrates of the modeling of input space by the hidden layer of an RBFNN. The midpoints of the clusters of data in input space form the weight vectors, $\boldsymbol{\mu}_i$, of the hidden nodes. The scatter of the data points of a cluster in input space determines the variance, σ_i^2 , of the Gaussians of the hidden nodes. The outputs of the hidden layer are passes to the final layer through linear basis function [35]. Fig.4.4 illustrates the class boundaries that are constructed by the first layer of an MLP. We summarize by noting that in RBFNNs the classification is done by using the closeness property of data in the same class, whereas, in MLPs the classification is done by difference between data.

4.4.2 Radial Basis Function Neural Networks for Pattern Mapping

In the case of pattern mapping/function approximation it is possible to obtain good generalization when many training examples are known throughout the range of the function. But this may not always be possible. Also due to presence of noise in the training data the problem may be ill-posed [18]. The method of *regularization* was proposed to overcome this problem. In this method a nonnegative functional, that makes use of prior knowledge about the function being approximated, is optimized along with the minimization of the risk functional [18], [56].

In the context of neural networks the introduction of the regularization term can be in the form of a smoothness constraint on the possible weight values. The magnitude of this extra term in the cost function governs the amount of smoothness applied to the surface being fit into the data points during the learning process [18].

The principle of *regularization* can be stated as follows: Find the function $f_{\mathbf{w}}$ that minimizes the cost function $\mathcal{E}(f_{\mathbf{w}})$, defined by

$$\mathcal{E}(f_{\mathbf{w}}) = \mathcal{E}_s(f_{\mathbf{w}}) + \lambda \mathcal{E}_c(f_{\mathbf{w}}) \quad (4.4)$$

where $\mathcal{E}_s(f_{\mathbf{w}})$ is the standard error term, $\mathcal{E}_c(f_{\mathbf{w}})$ is the regularization term, and λ is the regularization parameter. This regularization term can be used to incorporate the smoothness constraint into the network. The parameter λ can then be called the smoothness parameter.

4.5 Illustrations with Synthetic Data

In this section we give a comparative study of the performance of an RBFNN which is a network that incorporates knowledge about the problem and an MLP where there is no scope to incorporate prior knowledge about the problem. We consider performance on a classification task in section 4.5.1, and in section 4.5.2 we consider a function approximation task.

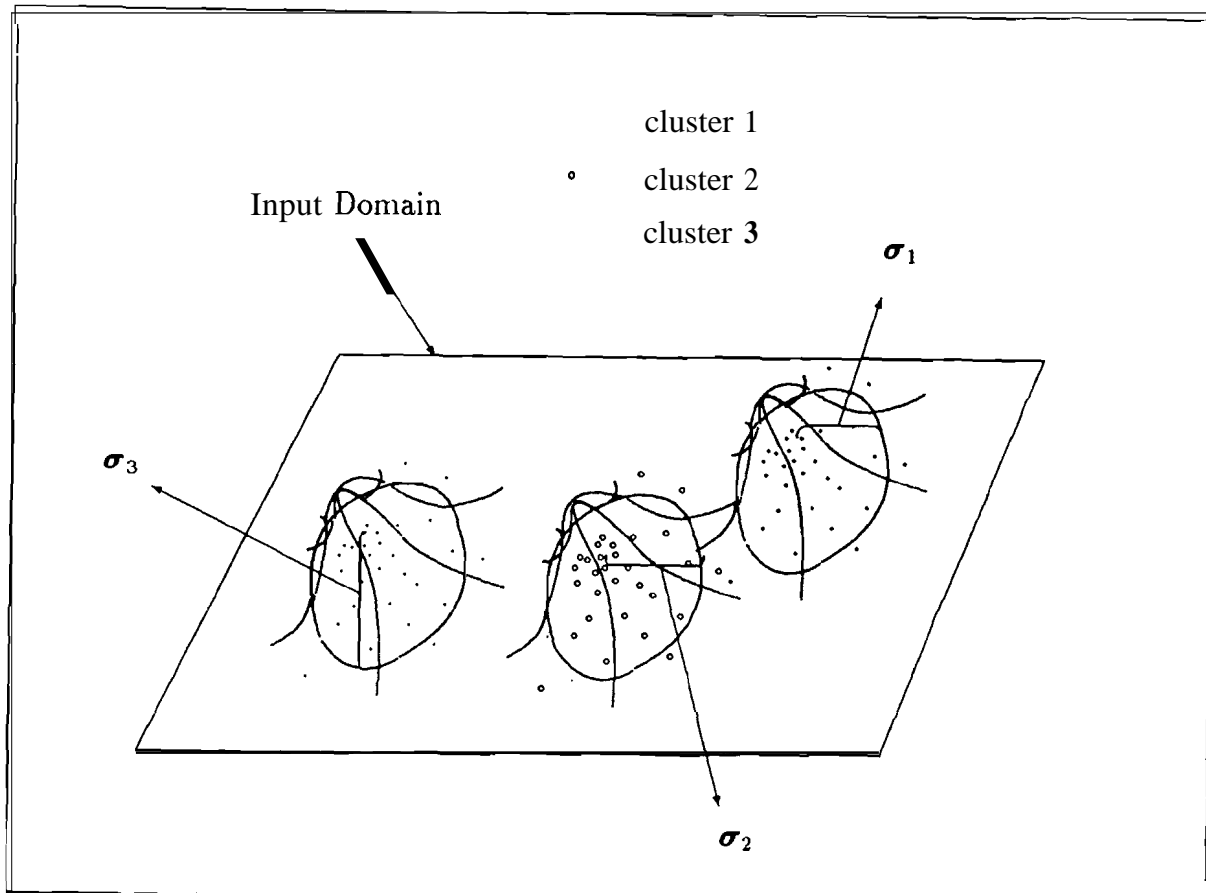


Figure 4.3: Illustration of the input space of the RBFNN as modeled by the input space. Here three clusters of data points in a 2 dimensional space are present. Each cluster is modeled by one hidden node of the RBFNN with "range of influence", σ , equal to the spread of the data points. The approximate center of the cluster forms the input weights, μ , to the hidden node.

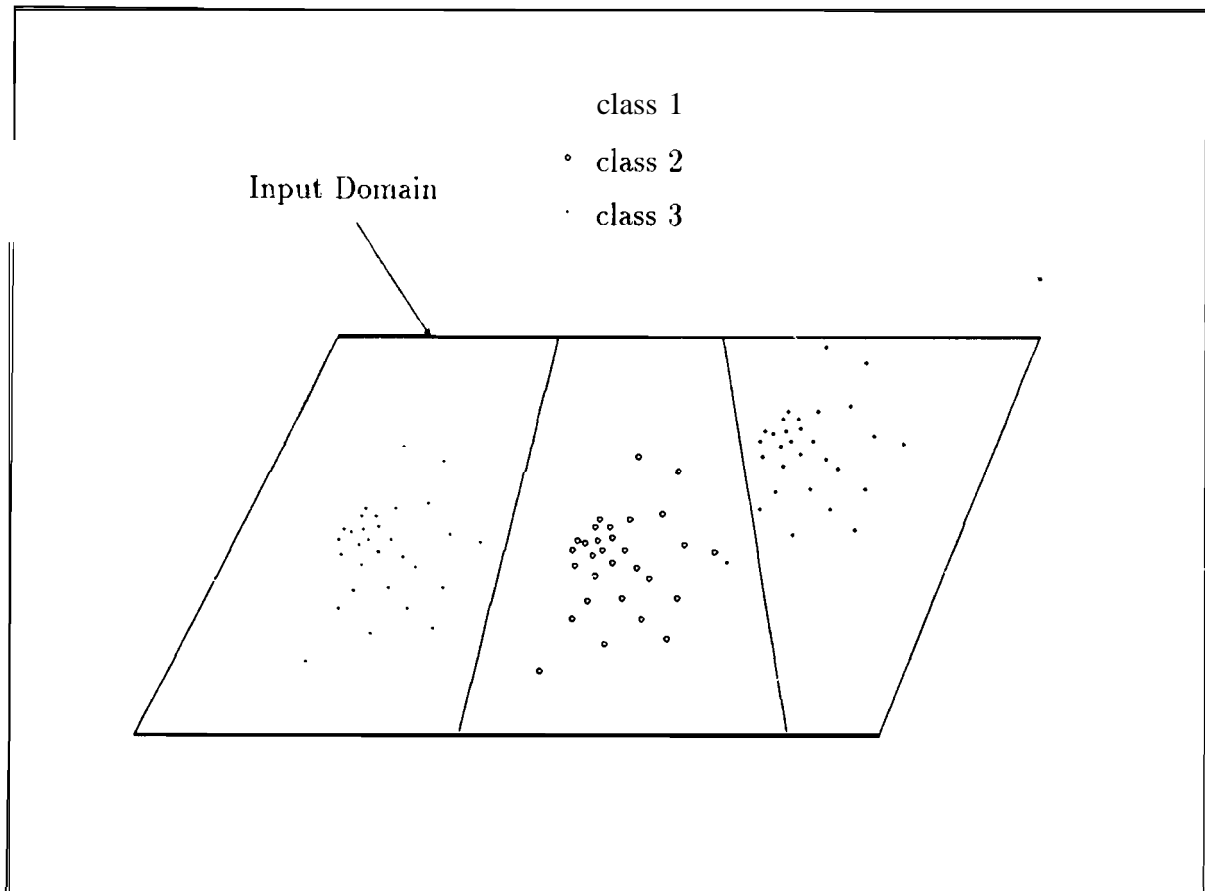


Figure 4.4: Illustration of the input space of a multilayer perceptron neural network as modeled by the input space. Here there are three clusters of data points in a 2 dimensional space. Each node in the first layer of the MLP realizes a straight line in input space that attempts to separate the clusters.

4.5.1 Classification with Prior Knowledge

In this case we consider classification of two dimensional data consisting of clusters of points. In the example considered, the input space has five clusters of points that are classified into three classes. Fig.4.5 illustrates the input data set.

Table 4.1 gives the generalization performance of the RBFNN on the classification problem. RBFNN of different sizes are trained on a training set consisting of 25 examples and the generalization performance is evaluated by considering the percentage correct classification of a test set consisting of 750 samples. It is observed from fourth row of the table that there is a steep increase in the performance of the network when the

number of nodes in the hidden layer becomes equal to the number of clusters of data in the input space. With further increase in the number of nodes in the hidden layer the generalization performance of the network remains high, unlike the cases generally discussed in the context of other network models where there is memorization when size of the network is large. Similar observations are made from table 4.2 which gives the performance of RBFNN when 100 examples are used to train it. It is inferred that the performance improves marginally because more data is available to perform the clustering at the hidden units.

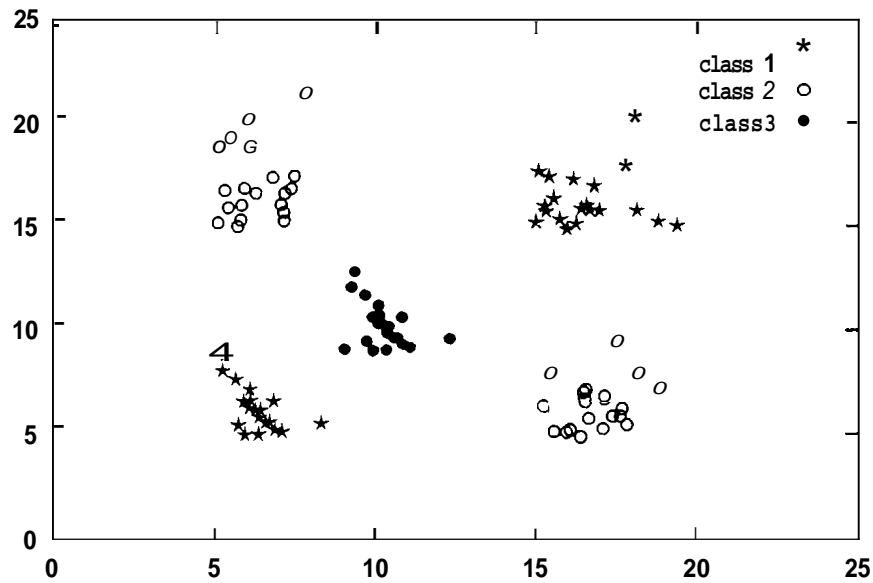


Figure 4.5: The set of 2-dimensional data used to compare the generalization capability of RBFNN and MLP for classification task. There are 5 clusters of data in the input space which are classified into 3 classes as shown.

Table 4.3 gives the generalization performance of different sizes of MLPs trained on 25 examples, and generalization is evaluated by performance on 750 test samples. It should be observed that the generalization performance is not as good as the RBFNN. Also, there is no specific trend exhibited in the variation of the generalization ability with increase in number of nodes. Table 4.4 gives the generalization performance of MLPs trained with 100 examples. It is found that the performance improves drastically when 7 hidden nodes are present in the network. But no specific reason can be given for this behavior. It should be noted that, there is no general trend in the way in which generalization capability is affected by increasing the number of hidden nodes.

Sl. No.	Number of Hidden Nodes	Percentage of Correct Classification
1	2	40.0%
2	3	60.3%
3	4	59.5%
4	5	90.0%
5	6	93.3%
6	7	92.8%
7	8	93.2%

Table 4.1: Performance of an RBFNN for classification on synthetic 2-dimensional data. The network is trained on 25 examples and tested on **750** samples.

Table 4.5 gives the performance of an MLP that has two layers of nodes. The choice of number of nodes is arbitrary, and uses the heuristic that the number of nodes in the second hidden layer should be equal to the number of clusters in the input space [19]. Apart from the above, we also apply the heuristic that the number of nodes in the second hidden layer should be half of the number of nodes in the first hidden layer. The generalization performance of such networks is tested. Although the generalization performance of two layered MLPs is good for combination of nodes given in rows **3; 6, 7, 9, 10** and 11 of table, yet no specific rule can be evolved for the choice of correct size of neural network. Though this does not seem to be a serious drawback in the case of this simple synthetic data, it is a major disadvantage when trying to design MLPs to solve real world problems. In many cases it may not be possible to arrive at the optimum size of a network by trial and error approach.

Thus, it can be inferred that in the case of RBFNN a systematic procedure of designing the network can be used to obtain good generalization. In the case when number of clusters in input space are known, we can expect good generalization, by choosing the number of hidden units equal to or greater than the number of clusters. Unlike the MLPs where the procedure for choice of number of hidden nodes in the neural network is arbitrary.

Sl. No.	Number of Hidden Nodes	Percentage of Correct Classification
1	2	40.0%
2	3	50.4%
3	4	61.8%
4	5	94.5%
5	6	94.5%
6	7	94.9%
7	8	97.2%

Table 4.2: Performance of an RBFNN for classification on synthetic 2-dimensional data. The network is trained on **100** examples and tested on **750** samples.

Sl. No.	Number of Hidden Nodes	Percentage of Correct Classification
1	2	75.6%
2	3	76.3%
3	4	76.0%
4	5	75.3%
5	6	75.5%
6	7	76.5%
7	8	76.4%

Table 4.3: Performance of an MLP for classification on synthetic 2-dimensional data. The network is trained on **25** examples and tested on **750** samples.

Thus, it can be concluded that inclusion of knowledge results in better generalization, when the size of the network chosen is sufficient for a given problem. Moreover, it allows systematic design of the networks.

Sl. No.	Number of Hidden Nodes	Percentage of Correct Classification
1	2	77.1%
2	3	77.2%
3	4	76.3%
4	5	66.4%
5	6	75.3%
6	7	96.5%
7	8	73.3%

Table 4.4: Performance of MLP for classification on synthetic 2-dimensional data. The network is trained on **100** examples and tested on 750 samples.

4.5.2 Function Approximation with Prior Knowledge

In this section we compare the generalization capability of the RBFNN and MLP. We consider the problem of approximating the function illustrated in the Fig.4.6. Different sizes of RBFNNs and MLPs are trained on training sets consisting of **50**, **100** and 200 training examples. The examples are generated by adding noise to the function evaluated at random points. Fig.4.7a gives training set of **50** examples and Fig.4.7b illustrates the training set of **100** examples. The performance is evaluated by plotting the function realized by the network and observing how similar it is to the original function.

Fig.4.8a shows the function obtained by an MLP with 5 hidden nodes trained on 50 examples. Thirty thousand iterations are required for it to converge. The variation of output error with number of iterations is given in the Fig.4.8b. Although it appears as a good approximation, by comparison with Fig.4.6 we observe that the approximation is poor when the input x varies from 4 to 8. Fig.4.9a shows the function obtained by an MLP with **10** hidden nodes trained on **50** examples. The variation of output error

Sl. No.	Number of Hidden Nodes $H1, H2$	Percentage of Correct Classification
1	5,2	77.1%
2	5,5	76.8%
3	6,3	97.1%
4	6,5	77.6%
5	7,3	77.1%
6	7,4	96.7%
7	7,5	96.7%
8	8,4	74.0%
9	10,5	98.4%
10	12,5	96.7%
11	12,6	97.2%

Table 4.5: Performance of the MLP for classification on synthetic 2-dimensional data. The network is trained on 100 examples and tested on 750 samples. The number of hidden nodes in the network is given by $H1, H2$, where $H1$ is number of nodes in first hidden layer, and $H2$ is number of nodes in second hidden layer.

with 11 training iterations is depicted in Fig.4.9b. The functions obtained by using 50 and 100 hidden nodes in the MLP are given in Fig.4.10a and Fig.4.11a, respectively. From these figures it can be observed that how the function approximation is better, and hence, generalization capability improves with increase in hidden nodes. But at the same time there are excessive variations in the function in some regions, for example, this is observed from Fig.4.11a when x varies in $[1, 3]$. Variation of the output error with training iterations for the MLPs with 50 hidden nodes and with 100 hidden nodes are given in Fig.4.10b and Fig.4.11b, respectively. It should be observed that the output error should be about 0.05 on the whole training set for convergence to occur. This is because the output varies between 0 to 1, and hence, error values should be less than 0.05 for the function to be approximated closely.

When 100 examples are used for training the network, it is found that the network does not converge to a low error value. Fig.4.12a shows the function the neural network with 100 hidden nodes realizes after thirty thousand training iterations. The function is very different from the actual function that is being approximated. From Fig.4.12b it is seen that the output error is more than 0.972 even after thirty thousand iterations. Fig.4.13a illustrates the function realized by a network with 150 hidden nodes trained on 100 examples. In this case we see that the function approximation is not as good as it should have been due to the availability of a large number of examples for training.

The functions realized by an RBFNN trained on 100 examples are shown in Fig.4.14a, Fig.4.15a and Fig.4.16a. These figures are obtained by choosing the smoothness parameter to be 0.3, 0.6 and 1.0, respectively. It is observed that the realized functions become closer to the actual function given in Fig. 4.6 with the use of higher smoothness parameter. Apart from this, it is observed from Fig 4.14b, Fig.4.15b and Fig.4.16b that the convergence to low error value takes place in a few iterations in the case of RBFNN. Unlike the case of MLP, in RBFNN there is no problem for convergence, and also fewer nodes are sufficient to realize the function. Fig.4.17a, Fig.4.18a and Fig.4.19a give the functions realized by RBFNN trained on 200 examples. These figures are obtained by using smoothness parameter 0.3, 0.6 and 1.0, respectively. It is observed from the Fig.4.17b, Fig.4.18b and Fig.4.19b that the convergence to low error value takes place within 100 iterations. The MLP does not converge to a low error value when trained on the training set with 200 examples. Table 4.6 summarizes the observations from Fig.4.8 to Fig.4.19.

Fig. No.	Observations	Remarks
4.8a, 4.9a, 4.10a, 4.11a	Curve obtained is smooth, but local variations in the function are not captured.	MLPs trained on 50 examples are able to generalize to some extent.
4.12a	When 50 training examples are used, the network with 100 nodes converges to a solution as observed from Fig.4.11a. But, it fails to converge for 100 training examples.	The generalization by the MLP with 100 hidden nodes and trained on 100 examples is poor.
4.13a	The network trained on 100 examples converges to a solution when the number of hidden nodes is increased from 100 to 150.	The generalization by MLP with 150 hidden nodes trained with 100 examples is good.
4.14a, 4.15a, 4.16a	RBFNNs converge faster than MLPs. When the smoothing factor is high, function approximation is better.	The generalization performed by RBFNN trained on 100 examples is good. The behavior of the function is captured well when many training examples are present.
4.17a, 4.18a, 4.19a	RBFNNs trained on 200 examples captures the behavior of the function well.	The generalization is good. Here the network converges to the solution, without getting stuck at local minima. The approximation of the function improves with increase in number of training examples.

Table 4.6: Summary of observations from Fig.4.8 to Fig.4.19 on comparison between MLP and RBFNN for function approximation.

Unlike the MLPs where the number of iterations required for convergence and the size of network increase, the RBFNNs converge to a solution even when the size of training set is large. Also, the variations of the function are captured better in the case of RBFNN. However, it should be noted that it is necessary to use a large number of examples in the case of RBFNN to achieve good approximation. When more examples are provided for training we achieve better approximation of the function, and this behavior is in fact similar to the example of modeling of system represented by data.

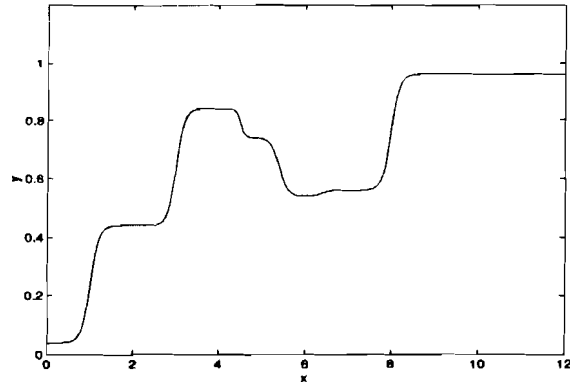


Figure 4.6: Function that is to be approximated by the neural networks.

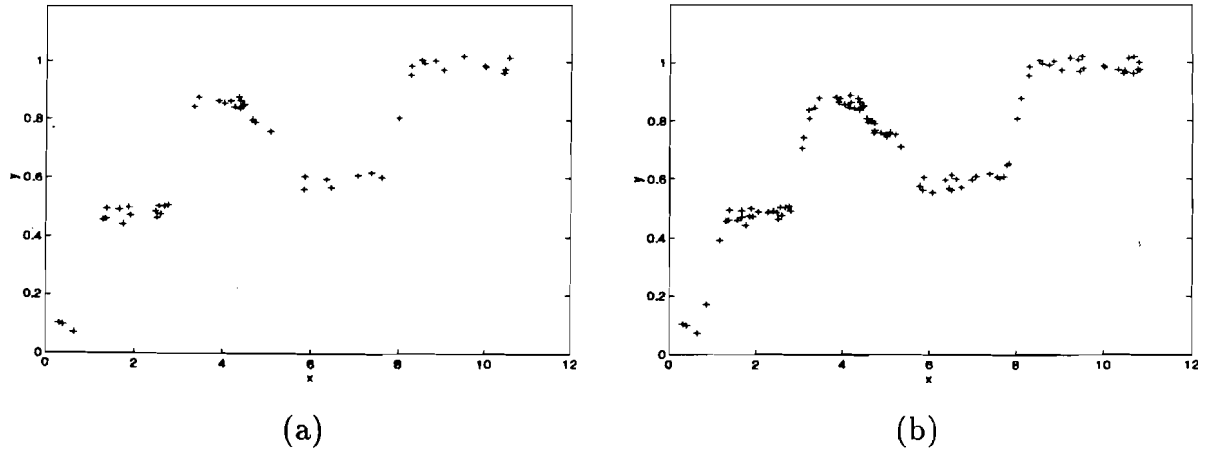


Figure 4.7: Set of training examples obtained by adding noise to the function evaluated at random points. (a) Training set of 50 examples. (b) Training set of 100 examples. Where each '+' indicates a training example. Different values of ' x ' are used as inputs to the network and corresponding values of ' y ' corrupted by noise are used as desired outputs.

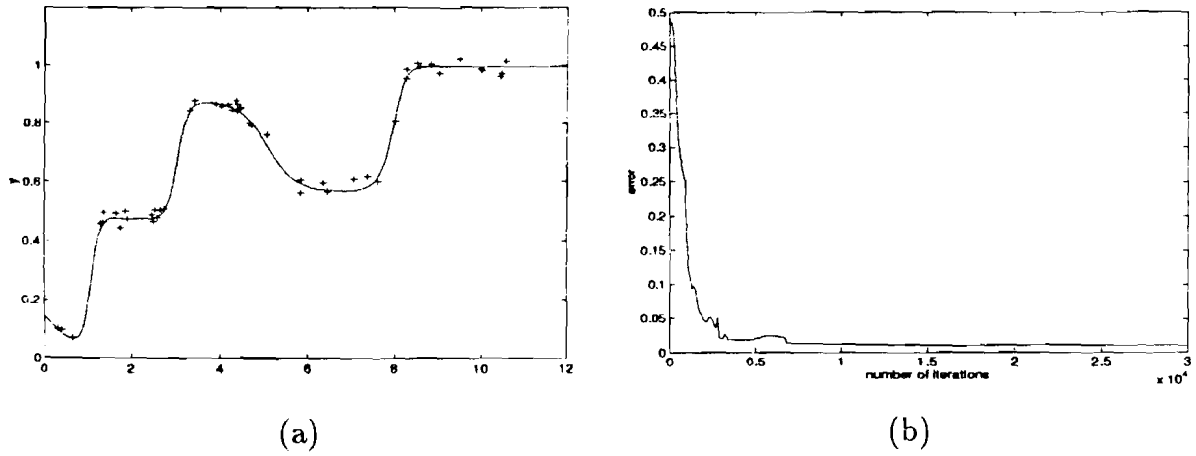


Figure 4.8: Function approximation by an MLP trained using backpropagation algorithm. Fifty training examples are used to train the network which has five hidden nodes. (a) Function realized by the network, where each '+' indicates a training example. (b) Graph of output error reduction with number of training iterations.

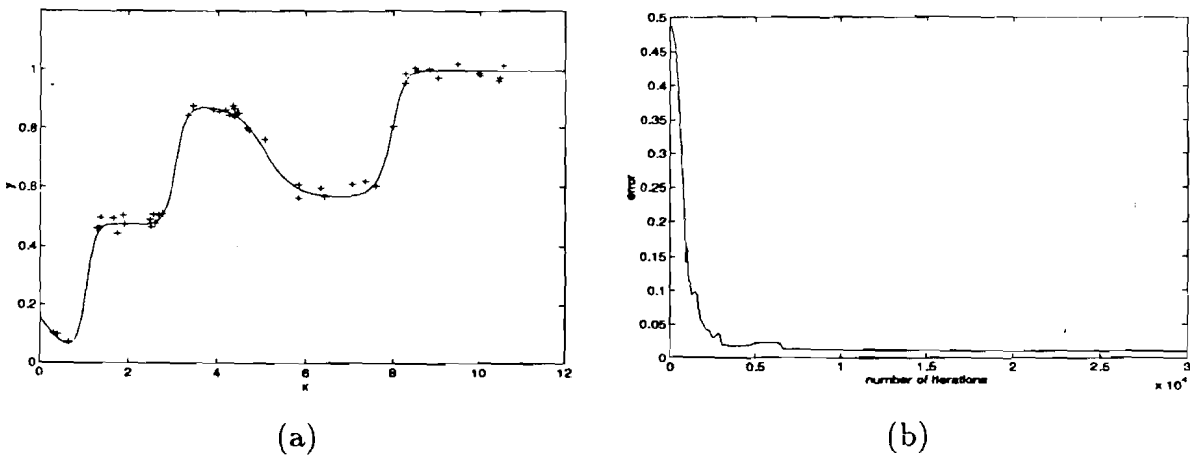


Figure 4.9: Function approximation by an MLP trained using backpropagation algorithm. Fifty training examples are used to train the network which has ten hidden nodes. (a) Function realized by the network, where each '+' indicates a training example. (b) Graph of output error reduction with number of training iterations.

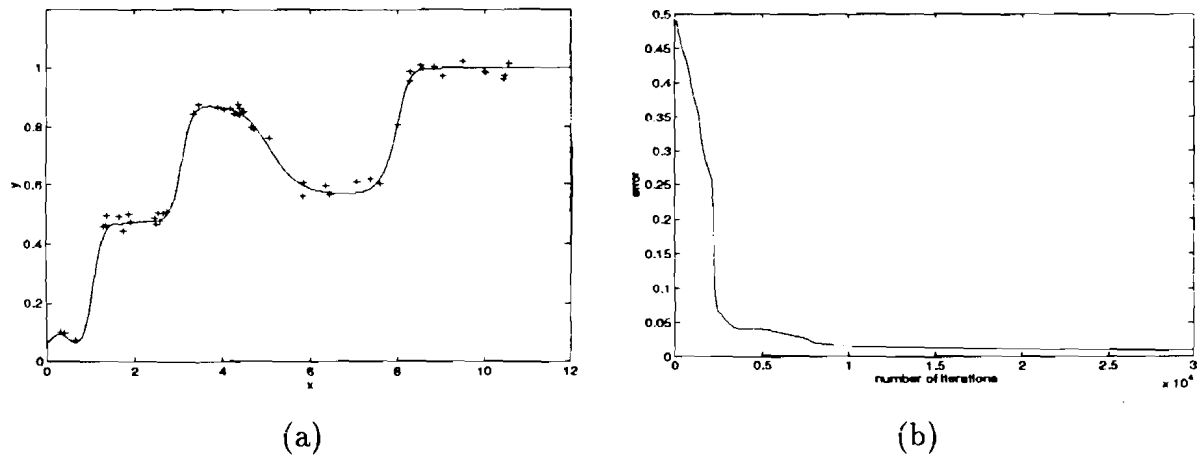


Figure 4.10: Function approximation by an MLP trained using backpropagation algorithm. Fifty training examples are used to train the network which has fifty hidden nodes. (a) Function realized by the network, where each '+' indicates a training example. (b) Graph of output error reduction with number of training iterations.

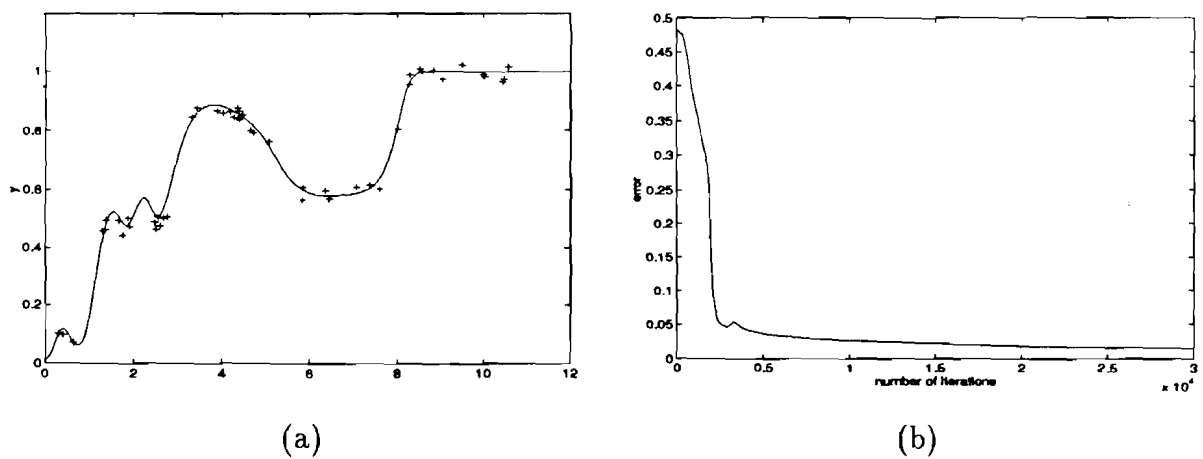
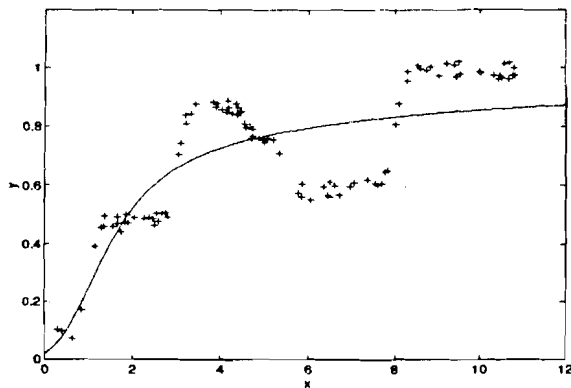
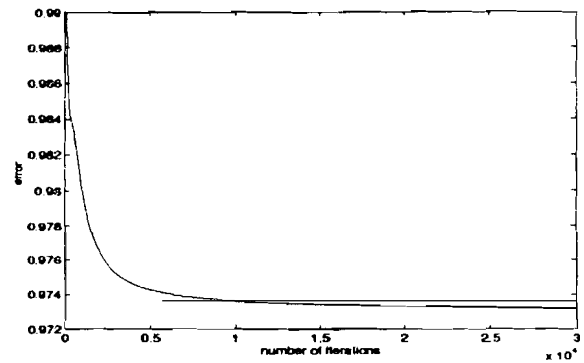


Figure 4.11: Function approximation by an MLP trained using backpropagation algorithm. Fifty training examples are used to train the network which has hundred hidden nodes. (a) Function realized by the network, where each '+' indicates a training example. (b) Graph of output error reduction with number of training iterations.

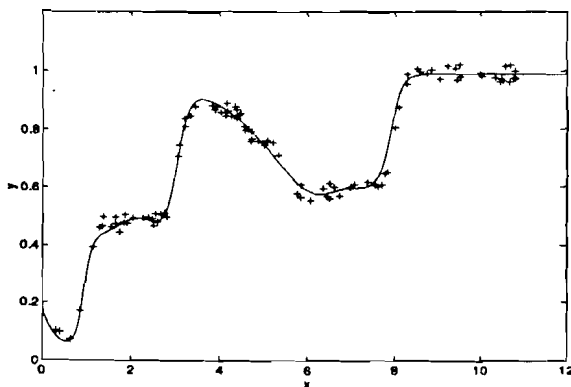


(a)

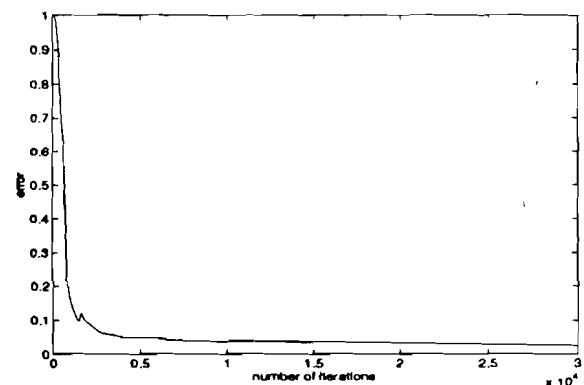


(b)

Figure 4.12: Function approximation by a MLP trained using backpropagation algorithm. Hundred training examples are used to train the network which has hundred hidden nodes. (a) Function realized by the network, where each '+' indicates a training example. (b) Graph of output error reduction with number of training iterations.



(a)



(b)

Figure 4.13: Function approximation by a MLP trained using backpropagation algorithm. Hundred training examples are used to train the network which has hundred and fifty hidden nodes. (a) Function realized by the network, where each '+' indicates a training example. (b) Graph of output error reduction with training iterations.

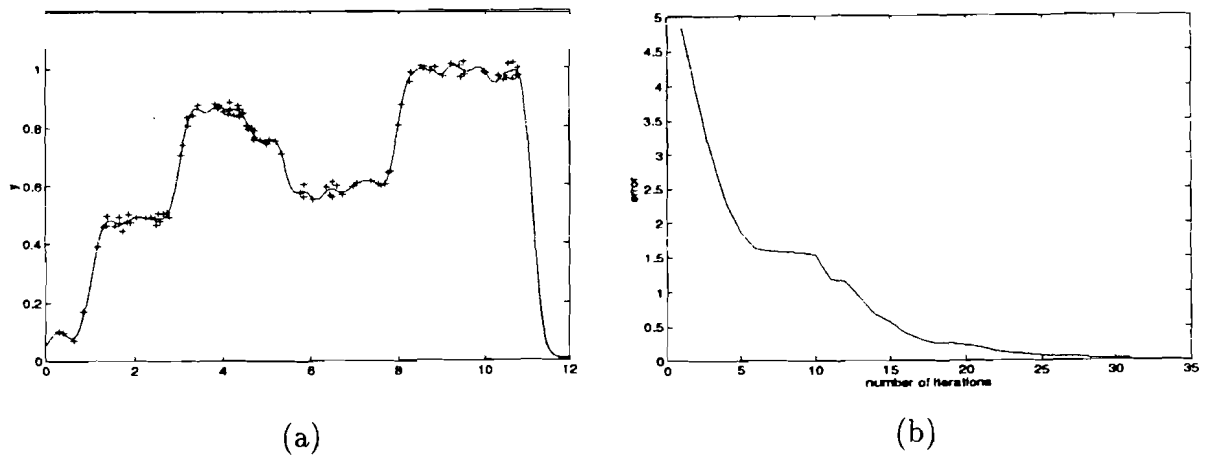


Figure 4.14: Function approximation by an RBFNN that has **34** hidden nodes and is trained on 100 examples. The value of the smoothing parameter used is **0.3**. (a) Function realized by the network, where each '+' indicates a training example. (b) Graph of error reduction with number of training iterations.

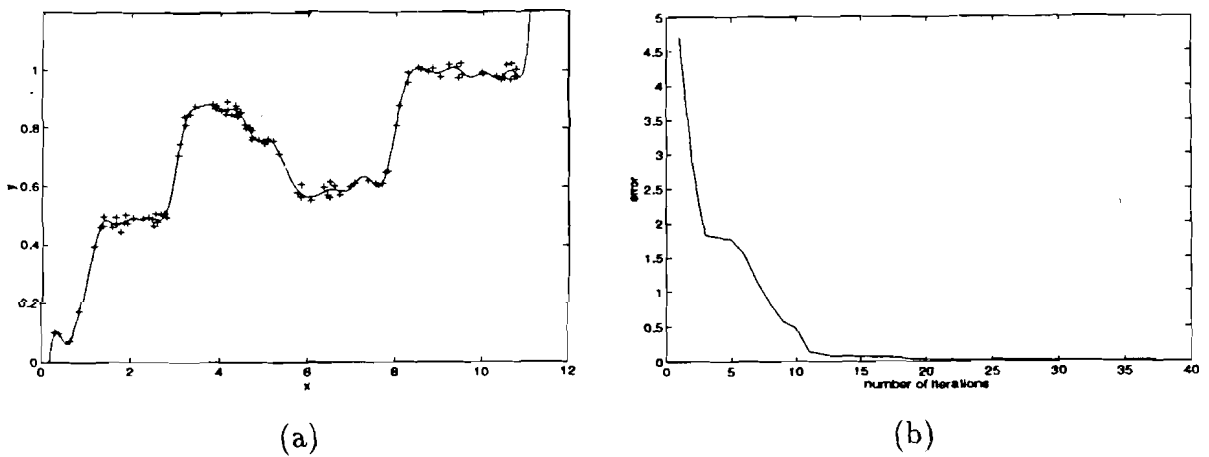


Figure 4.15: Function approximation by an RBFNN that has **38** hidden nodes and is trained on 100 examples. The value of the smoothing parameter used is **0.6**. (a) Function realized by the network, where each '+' indicates a training example. (b) Graph of error reduction with number of training iterations.

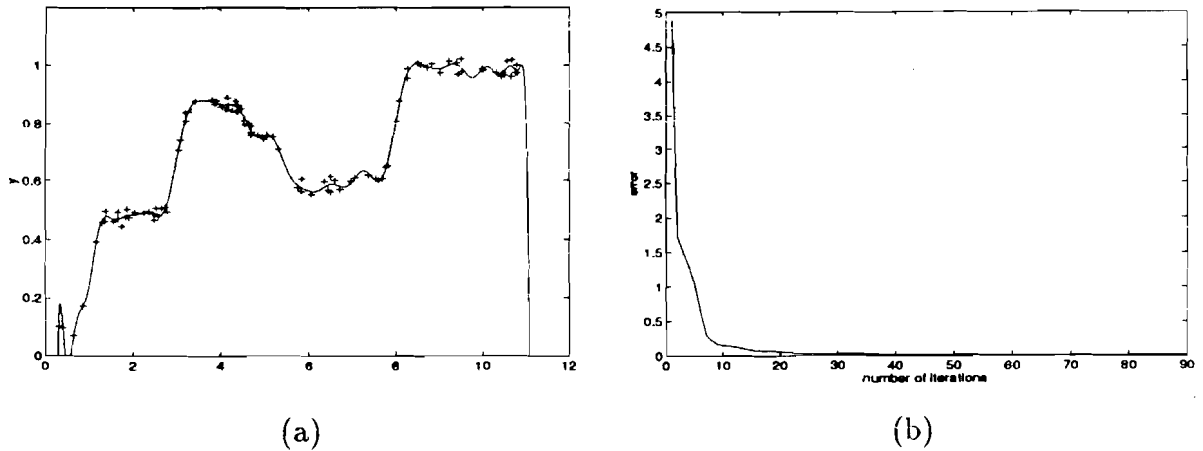


Figure 4.16: Function approximation by an RBFNN that has **90** hidden nodes and is trained on **100** examples. The value of the smoothing parameter used is **1.0**. (a) Function realized by the network, where each '+' indicates a training example. (b) Graph of error reduction with number of training iterations.

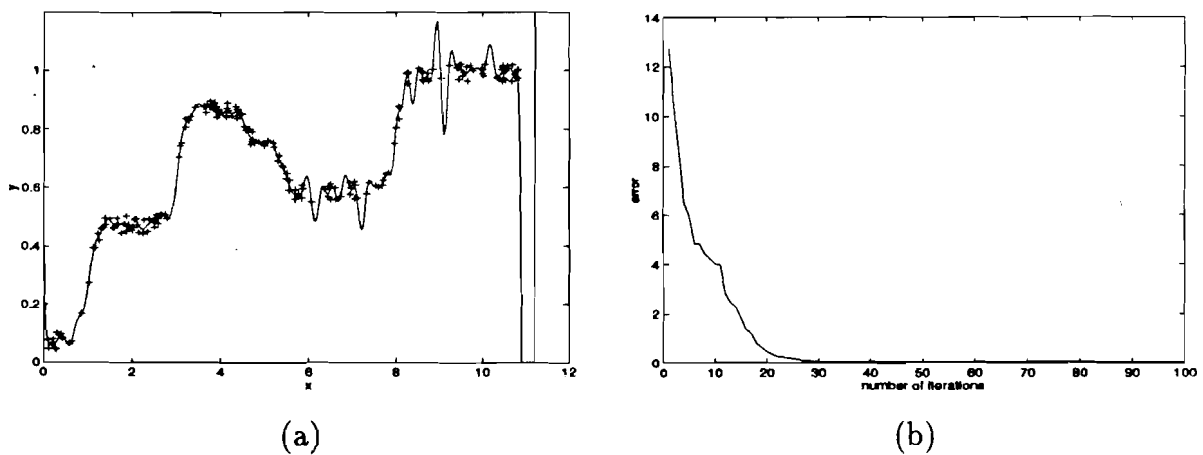


Figure 4.17: Function approximation by an RBFNN that has **100** hidden nodes and is trained on **200** examples. The value of the smoothing parameter used is **0.3**. (a) Function realized by the network, where each '+' indicates a training example. (b) Graph of error reduction with training iterations.

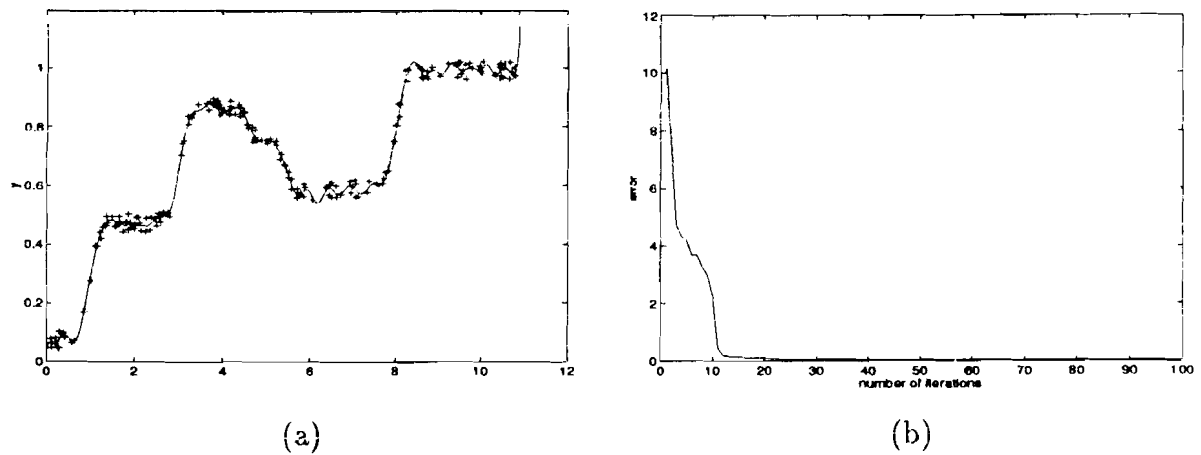


Figure 4.18: Function approximation by an RBFNN that has 100 hidden nodes and is trained on 200 examples. The value of the smoothing parameter used is 0.6. (a) Function realized by the network, where each '+' indicates a training example. (b) Graph of error reduction with training iterations.

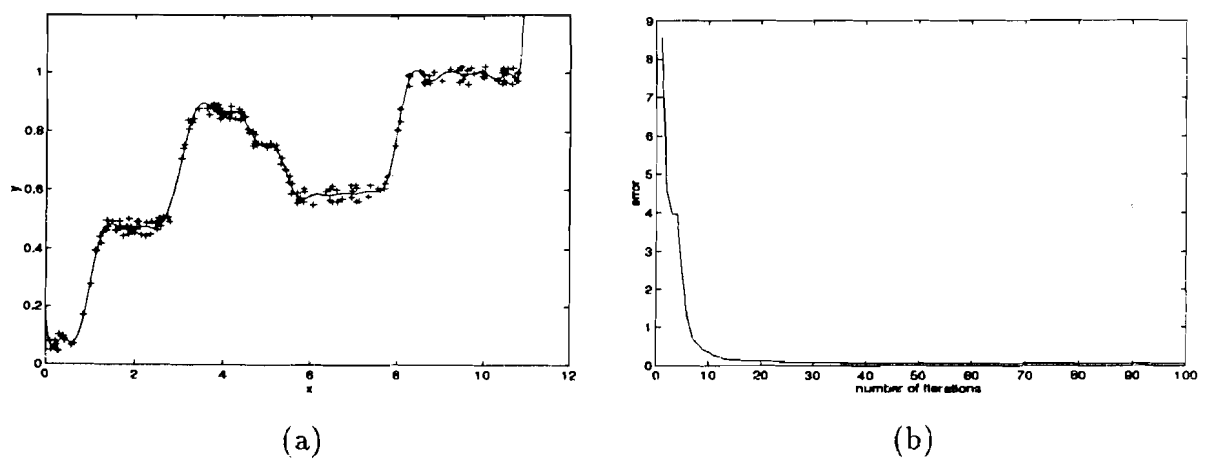


Figure 4.19: Function approximation by an RBFNN that has 100 hidden nodes and is trained on 200 examples. The value of the smoothing parameter used is 1.0. (a) Function realized by the network, where each '+' indicates a training example. (b) Graph of error reduction with training iterations.

4.6 Summary

in this chapter we discussed the problem-dependent nature of the generalization phenomenon.

An analogy with modeling of a system represented by data is presented, and extending this idea a concept for ideal behavior of generalization is provided where the problem of overtraining should not occur.

The advantages of incorporating problem-specific knowledge into the neural network is presented with respect to reduction in variance and bias of the function. We have also presented some models in which the knowledge of the problem is included, namely the RBFNN which is used for classification and pattern mapping tasks. Comparisons between the performance of RBFNNs and MLPs for pattern classification and function approximation problems were given using synthetic data. The advantages of inclusion of problem-dependent knowledge into the RBFNN is brought out. However, it is to be noted that the MLP does not always perform worse than the RBFNN, but there are certain kinds of problems in which the RBFNN gives significantly better results.

Chapter 5

ILLUSTRATIONS OF GENERALIZATION STUDIES WITH SPEECH DATA

5.1 Introduction

An important property of real world pattern recognition tasks is the presence of features in the data. Any data of pattern recognition tasks consists of features and redundant information. The choice of suitable features that represent patterns in the data and suppress the redundant information, determines the possibility to generalize. Thus, proper choice of features enable the neural network to generalize well. Selection of features to represent the data is problem dependent, and hence, it is necessary to adopt a problem dependent approach to improve generalization significantly.

In this chapter we illustrate the significance of a *priori* knowledge about the nature of data, in improving generalization by a neural network for real data. We consider the case of speech data, which is the output of a time varying vocal tract system excited by time varying excitation. For most speech recognition studies, the shape of the vocal tract system represents the type of sound being produced. But the shape information is embedded in the speech data through a complex transformation of the excitation into the speech signal. In general, speech signal is processed to extract some spectral parameters which reflect indirectly the vocal tract shapes. Since most of the time the extracted parameters from speech data use standard signal processing methods, the parameters represent primarily the signal information rather than the vocal tract system information. On the other hand, if we use a model for speech production and then extract the model parameters from speech, then the model represents our knowledge of the speech production. If the model is good, then the model parameters represent the system better. Therefore, for recognition studies, such as vowel recognition, generalization by the neural

network is better if the parameters represent the vocal tract system rather than the signal data. Since the vocal tract, features are embedded deep in the signal data, unless they are extracted explicitly the neural network cannot capture the vowel characteristics from the parameters representing the data only. The next section shows the significance of proper feature extraction for vowel classification.

We also illustrate the significance of feature representation for a pattern mapping task using speech data. We discuss the task of capturing the transformation of vocal tract systems between two speakers during production of continuous speech. It is shown that the transformation is captured effectively using features representing the vocal tract system.

Speech recognition is one of the most extensively studied pattern recognition tasks [43], [1]. It is a very challenging problem as data is naturally occurring and exhibits large variations in patterns, that is, the same words in speech can be uttered in several different ways. There is a need to capture features from a finite number of examples and use the features to recognize new patterns; therefore generalization capability is essential here. Speech exhibits embedded features, hence the effect of choice of features on generalization can be illustrated well with speech data.

Apart from this, speech recognition problems involve several types of pattern recognition tasks, namely, pattern clustering, pattern storage/retrieval, pattern classification and pattern mapping. This makes speech recognition problems interesting for studying pattern recognition tasks. In this work we concentrate on pattern mapping and pattern classification tasks.

5.2 Generalization in Vowel Recognition Task

5.2.1 The Vowel Classification Task

We consider the task of vowel recognition for the analysis of generalization in pattern classification. Vowels are speech sounds that are produced by a steady vocal tract system excited by the vibrating vocal cords. Different vowels are produced by changing the shape of the vocal tract. For our study we consider the vowels 'a', 'e', 'i', 'o' and 'u'. The data required for training is collected from vowel part of utterances of consonant

vowel pairs of three different speakers. The raw speech signal cannot be used directly for training the neural network because the features are deep hidden. Therefore, we extract features from the speech signal and use them for training the network. The goal of the experiment is to find out what kind of features are a better representation of the data for the classification task. For this purpose we consider formants and cepstral coefficients obtained from the utterances of speech as the extracted features. Formants are the resonances of the vocal tract and represent the characteristics of the system producing the signal. On the other hand, cepstral coefficients are obtained from the spectrum of the signal, and hence, represent characteristics of the signal more rather than the system.

We train a feedforward neural network using the backpropagation algorithm for realizing the classification. We train the network on the first three formants extracted from the speech signal of each utterance. The formants are extracted by taking the LPC and finding the frequencies at which the spectrum reaches peaks. The cepstral coefficients are calculated from the LPCs using the following formula [43]:

$$kc_k = -ka_k - \sum_{n=1}^{k-1} (k-n)c_{k-n}a_n \quad (5.1)$$

where a_k s are the LPCs and c_k s are the cepstral coefficients. In this work we have considered the first 12 cepstral coefficients of each utterance of vowels as input for classification.

We train neural networks of different sizes on the formants and cepstral data extracted from the utterances of various speakers. The neural network trained on formants has 3 inputs and 5 outputs. The target patterns are 5 dimensional vectors containing 1 in one location and 0 in all others. In the case of cepstral coefficients the network has 12 inputs and 5 outputs. The targets are similar to the targets used in formant classification. The neural networks are trained on 300 examples and tested on 1800 samples of test data to evaluate their generalization capability. The error rate measure is used to quantify the generalization capability. We discuss the results of the experiment in the following section.

5.2.2 Results and Discussion

Table 5.1 gives the classification performance of neural networks trained on formants. Table 5.2 gives the performance of different sizes of neural networks trained on cepstral

Sl. No.	Number of Hidden Nodes	Percentage Correct Classification of Test Set
1	40	88.8%
2	50	89.4%
3	60	89.4%
4	70	89.2%
5	80	88.7%
6	90	88.2%

Table 5.1: Results of vowel classification using fromant data for different sizes of networks.

coefficients. We find that the generalization performance is better in the case when formant data is used for training. Apart from this, it is found that the size of the network required for formant data is smaller and also the number of iterations required to converge to a given low error is less in the case of formant data compared to the case of cepstral data.

Thus, it is observed that generalization is better when features related to the vocal tract system are used. We demonstrate this using formants, the resonances of the vocal tract, to represent the speech information for vowel recognition task. The generalization is poor if parameters are not directly related to the vocal tract system. This shows the importance of considering problem-dependent information in order to achieve good generalization, as no amount of tuning of the parameters of the network can improve generalization when the input to the network does not represent the relevant feature in the context of the problem being solved.

5.3 Generalization in Voice Conversion

The pattern mapping tasks involve transformation of a function in input space to a function in output space. The input space and output space may be of high dimension and individual points in the space may exhibit features. Here we discuss some results from the

Sl. No.	Number of Hidden nodes	Percentage Correct Classification of Test Set
1	40	61.3%
2	50	58.3%
3	60	65.3%
4	70	62.4%
5	80	61.7%
6	90	63.2%

Table 5.2: Results of vowel classification using cepstral data for different **sizes** of networks.

problem of voice conversion [8] to bring out the importance of proper selection of feature representation in order to achieve good generalization.

Voice conversion involves transforming one speaker's voice into that of another speaker's voice. In any voice conversion system, it is necessary to capture the nonlinear vocal tract transformation between the two speakers using sample utterances from the speakers. The corresponding sound segments are taken in both the utterances, and a neural network is trained using information from segments of one speaker as input and that from the other speaker as output. It was shown [36] that if the information of the segments was represented using formants, then the complex nonlinear transformation of the vocal tract system could be captured effectively even for segments corresponding to dynamic situations not used in training. This study clearly illustrates the need for proper feature representation in order to achieve good generalization.

5.4 Summary

In this chapter we discussed the importance of using the correct features to represent the data in a pattern recognition task, in order to achieve good generalization. We have illustrated this by considering examples from speech recognition tasks.

We have addressed the issue of vowel recognition **as** an example of classification task.

We note that the generalization performance of the neural network is significantly better when **formants** are used as input data than when cepstral coefficients are used.

To illustrate generalization in the case of pattern **mapping** we discussed the case of voice conversion. In this case it is necessary to capture the vocal tract **transformation** between two speakers. It is found that this transformation of vocal tract can be captured well by using formants to represent the input data. Thus, in both cases the importance of using system related features rather than signal related features is brought out.

It is shown in this chapter that the choice of suitable data representation for a **given** task can result in significant improvement in generalization. Thus, the importance of adopting a problem-dependent approach to study and improve generalization is emphasized in this chapter.

Chapter 6

SUMMARY AND CONCLUSIONS

6.1 Summary and Conclusions

In this work we first described, at the conceptual level, generalization in the pattern clustering and pattern association tasks. We focused on pattern association and discussed the nature of some association tasks where generalization is possible and where generalization is not possible.

A brief overview of some existing measures of generalization were given. Some theoretical results were briefly reviewed. Theoretical studies of **generalization** consider an abstract model of learning from examples. In this kind of model synthetic data is used, and learning from examples is performed by minimizing an objective criterion. The objective criterion in general does not take into account the presence of features in the data.. This limits the applicability of theoretical results to real world pattern recognition problems.

Generalization in the context of feedforward neural networks was discussed. The feedforward neural networks are commonly used for learning pattern associations. Learning is implemented by minimization of an objective function. Generalization is measured by evaluating the performance of the network on a test set. For real data the use of the objective criterion leads to overtraining, and hence, lack of **generalization**. We discussed methods for improving generalization in the feedforward neural networks that involve: varying the number of parameters in the network, modifying the method of adjustment of parameters in the learning procedure and manipulating the training data.

We examined alternative methods for improving the **generalization** which included:

- Using a stopping criterion to overcome the problem of overtraining. Here we made use of the probabilistic interpretation of the outputs of the neural network. It is

realized by considering the result that the sum of a posteriori probabilities of all classes is unity.

- Modifying the methods of presentation of data using various block sizes of the training data in the block update method.
- Using two sets of training data, the first set is to train the network till near zero error, and the second set is to perturb the weights in order to overcome the problem of memorization.

We also discussed the limitations of the generalization measure adopted in these studies. Normally generalization is quantified by evaluating the performance of network on a test set using an objective measure. A measure called generalization index was introduced which takes into account the fuzzy nature of the concept of generalization. We illustrated the significance of this measure for some applications. We also brought out the limitations of the index due to the difficulty in measuring newness of a sample of test set.

We considered an analogy with modeling of system represented by data to state the desired feature of the generalization. Here the number of training samples merely determine the bias and variance of the estimated parameters of the model. Larger bias and variance result in poorer generalization. The question of overtraining does not arise in the modeling problems. Extending this concept to the case of neural networks, it was shown that it is necessary to incorporate the problem specific knowledge in the network and learning so that training with examples progressively captures the characteristics of the system with smaller bias and variance. We discussed these issues with special reference to pattern classification and pattern mapping tasks. We showed that incorporation of closeness property of features in the input patterns for each class results in improved generalization capability. Likewise taking into consideration the smoothness property of mapping function enhances the generalization capability of a network model, used for function approximation/pattern mapping.

We investigated the use of problem specific improvements in generalization capability of neural networks by considering speech data. Speech data was used because it is naturally occurring, exhibits features embedded in data, and speech recognition involves all pattern recognition tasks, namely, pattern clustering, storage/retrieval and classification/mapping. We focused on the classification and mapping tasks.

Recognition of isolated utterances of vowels was considered for pattern classification task. We showed that better generalization is possible when features related to the vocal tract system were used. We demonstrated this using formants, the resonances of the vocal tract to represent the speech information for vowel recognition task. The generalization was poor when parameters, not directly related to the vocal tract system, are used to train the network. This illustrated the problem dependent nature of the generalization phenomenon.

We examined the problem of transformation of the vocal tract shape from one speaker to another as an illustration of the pattern mapping task. Formants extracted from continuous speech of corresponding segments of two speakers were used to train a neural network. The resulting transformation was found to be continuous and natural even in the transition regions. Also, it was found that the performance was good for the vocal tract shapes not used in training the network.

Thus it can be concluded that, the phenomenon of generalization is dependent on the features present in the data for most pattern recognition tasks. Generalization capability cannot be studied independent of the problem being addressed. Theoretical results have to consider problem specific knowledge in order to be applicable to practical situations. In addition to this, methods of improving generalization need to be tailored to the problem being addressed to achieve significant improvement.

6.2 Suggestions for Future Research

While many of issues have been discussed in this thesis, many more challenging problems remain in the field of generalization for pattern recognition tasks. The following are certain problems that can be further investigated for improving generalization in feedforward networks:

- *Development of a systematic way of problem dependent design of neural networks:* Most existing approaches to obtain solutions to pattern recognition problems using neural networks are ad hoc and rely on trial and error methods to find a network that performs well. So there is a need to define a procedure for designing neural networks for specific tasks.

- *Development of continuous learning algorithm:* Most real world pattern recognition tasks involve a large variety of data and, it is not possible to collect all the data and train the network to take care of all possible situations. Therefore, methods that enable updation of weights when network is unable to give the target output are desirable.
- *Methods of problem dependent design of cost function:* As discussed: one of the major limitation of feedforward networks is that the cost function is not tailored to the problem. Hence, formulation of the cost function, which has to be minimized during the learning process taking into consideration knowledge of the problem, is an issue that can be further investigated.

Some the problems that can be investigated in the context of quantifying generalization measure are:

- *Alternative methods of combining individual generalization values of individual test samples:* In the proposed method of measuring generalization, the generalization value that we get from individual samples were combined using fuzzy set theoretic methods. Alternative methods using fuzzy measure theory [30] can be investigated.
- *Measure of newness of sample:* The newness of samples of a test set were calculated in the implementation of the proposed generalization index by considering Euclidian distance. This limits the scope of the measure. Other measures of newness can be investigated based on different distance concepts.

Appendix A

Proof of Convergence to A *Posteriori* Class Probability

In this appendix we show that by training through minimization of least square error results in the outputs converging to the a *posteriori* class probability $\pi(c|\mathbf{x})$ [39]. This result was used in proposing the alternative stopping criterion for training in section 3.4.1.

Let, $\pi(\mathbf{x}, c)$ be the underlying probability distribution of the training set $T_k = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_k, \mathbf{y}_k)\}$. The training criterion to be optimized is denoted by F and it depends on the mapping $f_{\mathbf{w}}$; therefore, we write $F(f_{\mathbf{w}})$ to express the dependence on $f_{\mathbf{w}}$. For the squared error criterion, we use $F(f_{\mathbf{w}})$ as a training criterion where $F(f_{\mathbf{w}})$ is expressed as follows:

$$F(f_{\mathbf{w}}) = \frac{1}{k} \sum_{i=1}^k \sum_{c=1}^C [t(\mathbf{y}_i, c) - f_{\mathbf{w}}(\mathbf{x}_i, c)]^2 \quad (\text{A.1})$$

where $t(\mathbf{y}_i, c)$ is the Kronecker delta function, used as a class indicator function that denotes the ideal target outputs and C is the number of classes. In the limit of large number of training samples, the sample average approximates the ensemble average. Thus, by taking the expectation over the joint distribution $\pi(\mathbf{x}, \mathbf{y})$, the training criterion $F(f_{\mathbf{w}})$ is expressed as,

$$F(f_{\mathbf{w}}) = \int d\mathbf{x} \sum_{\mathbf{y}} \pi(\mathbf{x}, \mathbf{y}) \sum_c [t(\mathbf{y}, c) - f_{\mathbf{w}}(\mathbf{x}, c)]^2 \quad (\text{A.2})$$

Note that the index \mathbf{y} stands for class membership, whereas, the index c denotes the output nodes. Here we want to understand what the output of the network $f_{\mathbf{w}}(\mathbf{x}, c)$ denotes. Therefore we interchange the sums over \mathbf{y} and c to obtain,

$$F(f_{\mathbf{w}}) = \int d\mathbf{x} \sum_c \sum_{\mathbf{y}} \pi(\mathbf{x}, \mathbf{y}) [t(\mathbf{y}, c) - f_{\mathbf{w}}(\mathbf{x}, c)]^2 \quad (\text{A.3})$$

$$= \int d\mathbf{x} \sum_c e(\mathbf{x}, c; f_{\mathbf{w}}(\mathbf{x}, c)) \quad (\text{A.4})$$

where $e(\mathbf{x}, c; f_{\mathbf{w}}(\mathbf{x}, c))$ is defined as the local error contribution at point \mathbf{x} and output, node c :

$$e(\mathbf{x}, c; f_{\mathbf{w}}(\mathbf{x}, c)) = \sum_{\mathbf{y}} \pi(\mathbf{x}, \mathbf{y}) [t(\mathbf{y}, c) - f_{\mathbf{w}}(\mathbf{x}, c)]^2 \quad (\text{A.5})$$

The local contribution $e(\mathbf{x}, c; f_{\mathbf{w}}(\mathbf{x}, c))$ is caused by the training samples from all the classes $y = 1, \dots, C$ and is thus obtained by weighting the squared error with the joint probability density function and summing over all classes $y = 1, \dots, C$. The sum over the index y can be rearranged by separating the index $y = c$, i.e., the correct output node with $t(\mathbf{y}, c) = 1$ from all other indices y with $t(\mathbf{y}, c) = 0$ and by considering the following relations:

$\pi(\mathbf{x}, c)$: density of the samples with desired output 1

$\pi(\mathbf{x}) - \pi(\mathbf{x}, c)$: density of the samples with desired output 0

Using the identity $\pi(\mathbf{x}, c) = \pi(\mathbf{x})\pi(c|\mathbf{x})$ we can rewrite the local error as,

$$\begin{aligned} e(\mathbf{x}, c; f_{\mathbf{w}}(\mathbf{x}, c)) &= \pi(\mathbf{x}, \mathbf{y}) [1 - f_{\mathbf{w}}(\mathbf{x}, c)]^2 + [\pi(\mathbf{x}) - \pi(\mathbf{x}, c)] [0 - f_{\mathbf{w}}(\mathbf{x}, c)]^2 \\ &= \pi(\mathbf{x}) [\pi(c|\mathbf{x}) [1 - f_{\mathbf{w}}(\mathbf{x}, c)]^2 + [1 - \pi(c|\mathbf{x})] [0 - f_{\mathbf{w}}(\mathbf{x}, c)]^2] \\ &= \pi(\mathbf{x}) [[\pi(c|\mathbf{x}) - f_{\mathbf{w}}(\mathbf{x}, c)]^2 + \pi(c|\mathbf{x}) [1 - \pi(c|\mathbf{x})]] \end{aligned}$$

The above function shows that the network output $f_{\mathbf{w}}(\mathbf{x}, c)$ is identical to the class probability $\pi(c|\mathbf{x})$, when the minimum value of the training criterion, F , is reached.

Appendix B

Overview of Some Fuzzy Set Concepts

In this appendix we give a brief overview of some fuzzy set concepts that are used in section 3.5 for proposing the fuzzy generalization index to quantify generalization [4].

B.1 Fuzzy Sets

In classical set theory, when a set of A is defined, any element of the universal set X can either be a member or not be a member of the set. This property of whether or not an element x of universal set belongs to set, A can be defined by a function μ_A . This function takes the value 1 if the element belongs to the set A and 0 if the element does not belong to it. This function is known as characteristic function. Therefore, for the set A , the characteristic function $\mu_A : X \rightarrow \{0, 1\}$ is given by

$$\mu_A(x) = \begin{cases} 1 & : \text{if and only if } x \in A \\ 0 & : \text{if and only if } x \notin A \end{cases}$$

But in many real life situations it is uncertain whether an element belongs totally to the set A , i.e., the element may belong to the set with a certain degree of belongingness. Fuzzy set theory has been developed in order to take care of such situations. In a fuzzy set the characteristic function of a set A can take values in $[0, 1]$. This function is known as membership function, because larger value of the function denotes more membership of the element to the set under consideration. Thus, the membership function μ_A is expressed as

$$\mu_A : X \rightarrow [0, 1]$$

Therefore, any concept that uses fuzzy sets requires the membership function to be defined. This function is usually designed by taking into consideration the requirements and constraints of the problem.

B.2 Aggregation Operator

An aggregation operator is defined by a function,

$$h : [0, 1]^n \rightarrow [0, 1] \quad (\text{B.1})$$

for some $n \geq 2$. One class of aggregation operators consists of generalized means. These are defined by the formula,

$$h_\alpha(a_1, a_2, \dots, a_n) = \left(\frac{a_1^\alpha + a_2^\alpha + \dots + a_n^\alpha}{n} \right)^{1/\alpha} \quad (\text{B.2})$$

where $\alpha \in \mathbb{R} (a \neq 0)$. The parameter a can be used to control the softness of the operator. This operator satisfies the following properties:

1. *Commutative* : $h(a, b) = h(b, a);$
2. *Associative* : $h(h(a, b), c) = h(a, h(b, c));$
3. $h(a, b) \leq h(c, d)$ when $a \leq c$ and $b \leq d$;

Appendix C

Experimental Observations for Generalization Index

In this appendix we present some observations of the experimental studies of the generalization index proposed in section 3.5.

The behavior of the generalization index proposed is studied by applying to the problem of Opening Bid in Contract Bridge game. Here the problem is to train a neural network to give the same bid as a human bidder for a given hand. Input of the network, is a hand pattern of a player, the pattern is represented as a 52 dimensional binary pattern, where a '1' represents the presence and a '0' represents the absence of a card. The input layer of the network consists of 52 nodes. The target output of the network is the first. level opening bid corresponding to the input hand. A fully connected three layer feedforward network with backpropagation [48] learning is used here. Fifty hidden nodes and five output nodes (corresponding to the bids: 1 Club, 1 Diamond, 1 Heart., 1 Spade and 1 Notrump) are used in this network. The network is trained with a set of input(hand)-output(bid) pairs. In this experiment, the Standard American bidding convention is used. The data for this experiment was collected from an Open Bridge Tournament.

In this experiment, we determine the newness of a test input (x_i) by finding its Euclidian distance from the nearest training input example which has the same output bid. This normalized distance lies between 0 and 1; and hence it is treated as the membership value for *newness*. This membership value is transformed to another membership value to signify *more or less new* by using a fuzzy hedge operator. This transformation is required to reduce the influence of newness of the example on y_i . Here, we choose the value of β (the parameter in hedge operator) and α (the parameter in the aggregation operator) as 2 and 16, respectively. We compare our measure with the most commonly

used error rate measure. Table C.i gives the generalization value obtained by running the network on various test sets, once it is trained on a fixed training set,. Observe that the value of fuzzy generalization index, \mathcal{G} , varies less than the value of the er which is (1-Error Rate), as it can be seen from the values of variance (σ^2) given in the last row of the table. Figure C.1 shows a graph of variation of generalization error with increase in number of training examples used to train the network. Here, it is observed that, \mathcal{G} reduces with increase in the number of training examples reflecting the fact that scope for generalization decreases. Note that the error rate measure does not take into account the decrease in scope for generalization due to increase in number of training examples.

Data Set	er	\mathcal{G}
1	0.680	0.747
2	0.840	0.733
3	0.780	0.744
4	0.780	0.736
5	0.620	0.756
6	0.780	0.742
7	0.720	0.731
8	0.800	0.741
9	0.760	0.753
10	0.720	0.708
σ^2	0.0641	0.0135

Table C.1: Comparison of Generalization Index, \mathcal{G} , with (1-Error Rate), er , for different test sets. The last row of the table gives the value of variance of measure over the different test sets.

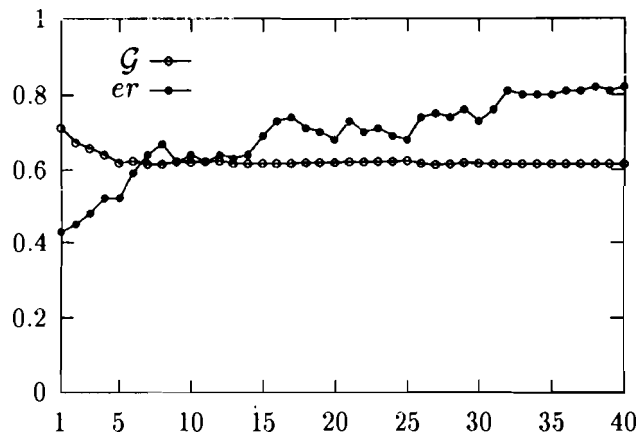


Figure C.1: Graph of variation of generalization error with increase in training examples. \mathcal{G} indicates the proposed generalization index and er denotes (1-Error Rate).

References

- [1] W. A. Ainsworth, *Speech Recognition By Machine*, LEE computing series 12. Peter Peregrinus Ltd., on behalf of the Institution of Electrical Engineers, London, United Kingdom, 1 edition, 1988.
- [2] Martin Anthony and Sean B. Holden, "Quantifying generalization in linearly weighted neural networks", *Complex Systems*, vol. 8, pp. 91–144, 1994.
- [3] Eric B. Baum and David Haussler, "What size net gives valid generalization?". *Neural Computation*, vol. 1, pp. 151–160, 1989.
- [4] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981.
- [5] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth, "Learnability and Vapnik-Chervonenkis dimension"? *Journal of the Association, of Computing Machinery*, vol. 36, no. 4, pp. 929–965, October 1989.
- [6] Sing-Tze Bow, *Pattern Recognition*, Marcel Dekker, Inc, New York, 1984.
- [7] Joachim M. Buhmann, "Data clustering and learning", in *The Handbook of Brain Theory and Neural Networks*, Michael A. Arbib, Ed., pp. 278–282. The MIT Press, 1995.
- [8] D. G. Childers, K. Wu, D. M. Hicks, and B. Yegnanarayana, "Voice conversion", *Speech Communication*, vol. 8, pp. 147–158, 1989.
- [9] D. Diankov, H. Hellendorn, and M Reinfrank, *An introduction to fuzzy control*, Springer-Verlag, Berlin, 1993.
- [10] J. Dombi, "Membership function as an evaluation", *Fuzzy Sets and Systems*, vol. 35, pp. 1–21, 1990.
- [11] R. C. Dubes and A. K. Jain, *Algorithms that Cluster Data*, Prentice Hall, Englewood Cliffs, NJ, 1987.
- [12] D. Dubois and H. Prade, *Fuzzy Sets and Systems*, Academic Press, New York, 1980.
- [13] Richard O. Duda and Peter E. Hart, *Pattern Classification and Scent Analysis*, vol. 1, John Wiley and Sons. 1973.
- [14] Kunihiro Fukushima and Nobuaki Wake, "Handwritten alphanumeric character recognition by the neocognitron", *IEEE Transactions on Neural Networks*, vol. 2, no. 3, pp. 355–365, May 1991.

- [15] D. Haussler, N. Littlestone, and M. K. Warmuth, "Predicting $\{0,1\}$ -functions on randomly drawn points", UCSC-CRL-90-54, University of California, Computer Research Laboratory, Applied Sciences Building. University of California, Santa Cruz, CA 95064. December 1990.
- [16] David Haussler, "Decision theoretic generalizations of the PAC model for neural net and other learning applications", *Information and Computation*, vol. 100, pp. 78–150, 1992.
- [17] David Haussler, Micheal Kearns, and Robert Schapire, "Bounds on the sample complexity using information theory", *Machine Learning*, vol. 14, pp. 83–113, 1993.
- [18] S. Haykin, *Neural Networks - A Comprehensive Foundation*, Macmillan college publishing company, Inc., 1 edition, 1994.
- [19] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*, vol. 1, Addison Wesley, 1991.
- [20] John Hertz, "Computing with attractors", in *The Handbook of Brain Theory and Neural Networks*, Michael A. Arbib, Ed., pp. 230–234. The MIT Press, 1995.
- [21] Sean B. Holden, *On the Theory of Generalization and Self-Structuring in Linearly Weighted Connectionist Networks*, PhD thesis, Corpus Christi College, Cambridge University Engineering Department. Trumpington Street, Cambridge CB2 1PZ, January 1994.
- [22] Sean B. Holden and Mahesan Niranjan, "On the practical applicability of VC dimension bounds", Tech. Rep. CUED/F-INFENG/TR.155, University of Cambridge. Cambridge University Engineering DEpartment, Trumpington Street., Cambridge CB2 1PZ, October 1994.
- [23] Sean B. Holden and Peter J. W. Rayner, "Generalization and PAC learning: Some new results for the class of generalized single layer networks", *IEEE Transactions on Neural Networks*, vol. 6, no. 2, pp. 368–380, March 1995.
- [24] Lasse Holmstorm and Petri Koistnen, "Using additive noise in back propagation training", *IEEE Transactions on Neural Networks*, vol. 3, no. 1, pp. 24–38. January 1993.
- [25] D. R. Hush and B. G. Horne, "Progress in supervised neural networks", *IEEE Transactions on Signal Processing*, pp. 8–39, January 1993.
- [26] Shun ichi Amari, "A universal theorem on learning curves". *Neural Networks*, vol. 6, pp. 161–166, 1993.
- [27] Shun ichi Amari, "Learning and statistical inference", in *The Handbook of Brain Theory and Neural Networks*, Michael A. Arbib, Ed., pp. 522–526. The MIT Press, 1995.

- [28] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, Englewood Cliffs, New Jersey, 1988.
- [29] Jack S. N. Jean and Jin Wang, "Weight smoothing to improve network generalization", *IEEE Transactions on Neural Networks*, vol. 5, no. 5, pp. 752–763, September 1994.
- [30] G. S. Klir and T. A. Folger, *Fuzzy Sets Uncertainty and Information*, Prentice-Hall, Englewood Cliffs, NJ, 1993.
- [31] S. Y. Kung, *Digital Neural Networks*, Prentice Hall, Englewood Cliffs, New Jersey, 1993.
- [32] Yann LeCun and Yoshua Bengio, "Pattern recognition", in *The Handbook of Brain Theory and Neural Networks*, Michael A. Arbib, Ed., pp. 711–715. The MIT Press, 1995.
- [33] Richard P. Lippmann, "Introduction to computing with neural nets", *IEEE ASSP Magazine*, pp. 4–22, April 1987.
- [34] Yong Liu, "Unbiased estimate of generalization error and model selection in neural network". *Neural Networks*, vol. 8, no. 2, pp. 215–219, 1995.
- [35] David Lowe, "Radial basis function networks", in *The Handbook of Brain Theory and Neural Networks*, Michael A. Arbib, Ed., pp. 779–782. The MIT Press, 1995.
- [36] Narendranath M, "Transformation of vocal tract characteristics for voice conversion using artificial neural networks", Master's thesis, Department of Computer Science and Engineering, Indian Institute of Technology, Madras 600036, November 1995.
- [37] John Makhoul, "Linear prediction: A tutorial review", *Proceedings of IEEE*, vol. 63, no. 4, pp. 561–580, April 1975.
- [38] M. T. Musavi, K. H. Chan, D. M. Hummels, and K. Kalantri, "On the generalization ability of neural network classifiers", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, June 1994.
- [39] Hermann Ney, "On the probabilistic interpretation of neural network classifiers and discriminative training criteria", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 2, pp. 107–119, February 1995.
- [40] S. K. Pal and D. Dutta Majumder, *Fuzzy Mathematical Approach to Pattern Recognition*, Wiley (Halsted Press), New York, 1986.
- [41] Tomaso Poggio and Federico Girosi, "Neural networks for approximation and learning". *Proceedings of IEEE*, vol. 78, no. 9, pp. 1481–1497, September 1990.
- [42] L. R. Rabiner and B. H. Juang, "An introduction to hidden markov models", *IEEE ASSP Magazine*, vol. 3, pp. 4–10, 1986.

- [43] L. R. Rabiner and B. H. Juang, *Fundamentals of Speech Recognition*, Printice-Hall, 1993.
- [44] Lawrence R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition", *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, February 1989.
- [45] P. P. Raghu, *Artificial Neural Network Models For Texture Analysis*, PhD thesis, Department of Computer Science and Engineering, Indian Institute of Technology, Madras 600036, November 1995.
- [46] A. Ravichandran and B. Yegnanarayana, "Studies on object recognition from degraded images using neural networks", *Neural Networks*, 1995.
- [47] Russell Reed, "Pruning algorithms - a survey", *IEEE Transactions on Neural Networks*, vol. 4, pp. 740–747, September 1993.
- [48] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation", in *Parallel and Distributed Processing*, D. E. Rumelhart and McClelland, Eds. MIT Press, Cambridge, MA, 1986.
- [49] H. S. Seung, H. Sompolinsky, and N. Tishby, "Statistical mechanics of learning from examples", *Physical Review A*, vol. 45, no. 8, pp. 6056–6091, April 1992.
- [50] Eduardo D. Sontag, "Feedforward nets for interpolation and classification", *Journal of Computing System Sciences*, vol. 45, pp. 20–48, 1992.
- [51] Hector J. Sussmann, "Uniqueness of the weights of minimal feedforward nets with given input-output map", *Neural Networks*, vol. 5, pp. 589–593, 1992.
- [52] H. Takagi and M. Sugeno, "Fuzzy identification of systems and applications to modeling and control", *IEEE Transactions on System, Man and Cybernetics*, vol. 15, pp. 116–132, January/February 1985.
- [53] J. Tau and R. Gonzalez, *Pattern recognition principles*, Addison Wesley, Reading, MA, 1974.
- [54] L. G. Valiant, "A theory of the learnable", *Communications of the ACM*, vol. 27, no. 11, pp. 1134–1142, November 1984.
- [55] V. Vapnik, "Learning and generalization: Theoretical bounds", in *The Handbook of Brain Theory and Neural Networks*, Michael A. Arbib, Ed., pp. 516–522. The MIT Press, 1995.
- [56] Grace Wahba, "Generalization and regularization in nonlinear learning systems", in *The Handbook of Brain Theory and Neural Networks*, Michael A. Arbib, Ed., pp. 426–430. The MIT Press, 1995.
- [57] R. S. Weng and M. R. Dudley, "Some special Vapnik-Chervonenkis classes", *Discrete Mathematics*, vol. 33, pp. 313–318, 1981.

- [58] B. Yegnanarayana, "Artificial neural networks for pattern recognition", *Sadhana*, vol. 19, no. 1, pp. 147–169, April 1994.
- [59] B. Yegnanarayana, Deepak Khemani, and Manish Sarkar, "Neural networks for contract bridge bidding", *Sadhana*, vol. 21, June 1996.

असतो मा सद्गमय, तमसो मा ज्योतिर्गमय
मृत्योर्मा अमृतं गमय ।

असतः - from untruth, सद् - truth, तमसो - from darkness, ज्योति - light, मृत्यो - from death, अमृत - immortality, गमय - lead (us).

Lead (us) from untruth to truth, from darkness to light, from death to immortality.

ॐ पूर्णभिदं पूर्णमिदं पूर्णात्पूर्णमुदच्यते ।
पूर्णस्य पूर्णमादाय पूर्णमेवावशिष्यते ॥

ॐ - auspicious sound, पूर्ण - whole, अदः - that (God), पूर्ण - whole, इद - this (world), पूर्णात् - from that whole, पूर्ण - this whole, उदच्यते - manifests, पूर्णस्य - of this whole, पूर्ण - wholeness, आदाय - retaining, पूर्ण - whole, एव - alone, अवशिष्यते - remains (ever).

That (God) is whole. This (world) is whole. From that whole this whole manifests. Retaining the wholeness of this whole that whole ever remains.

or

That (God) is unmanifest Brahman. This (the world) is manifest Brahman. From that unmanifest Brahman arises this manifest Brahman ie world is manifest God. The unmanifest Brahman ever remains the same as the heart of the constantly changing manifest world.