

**Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596**

SYSTEM IDENTIFICATION USING NEURAL NETWORKS

H.-S. Dai

**January 1993
CSD-930002**

UNIVERSITY OF CALIFORNIA

Los Angeles

System Identification Using Neural Networks

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Computer Science

by

Han-Sen Dai

1992

© Copyright by

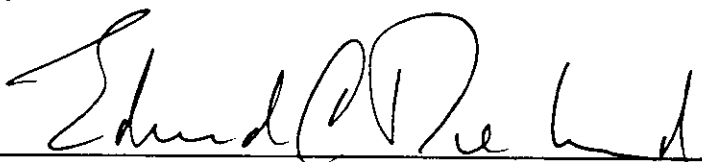
Han-Sen Dai

1992


The dissertation of Han-Sen Dai is approved.



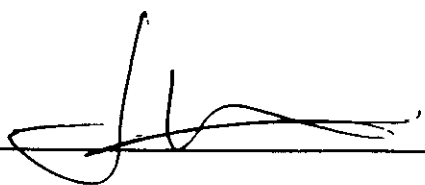
Jack W. Carlyle



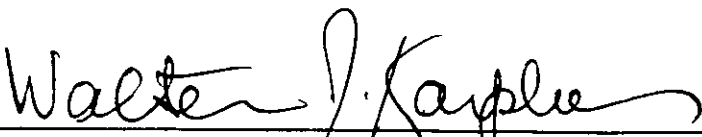
Edward C. DeLand



Nhan N. Levan



Jacques J. Vidal



Walter J. Karplus, Committee Chair

University of California, Los Angeles

1992

To my father, *Chin-Rong Dai*, and
to the memory of my mother, *Siue-Lien Hwang*,

whose encouragement and support gave me confidence,
whose values influenced my choice.

TABLE OF CONTENTS

1	Introduction	1
1.1	Objectives	2
1.2	Methodology	3
1.3	Organization of Dissertation	4
I	Background	6
2	Problem Formulation	7
2.1	Introduction	7
2.2	System Identification	7
2.3	Related Work	11
2.3.1	Optimization Approach	11
2.3.2	Pattern Recognition Approach	12
2.4	Problem Statement	13
2.5	Why is Pattern Recognition Approach Important to System Identification Problem?	14
3	Neural Networks	17
3.1	Introduction	17
3.2	Definition and Attributes	17
3.3	Motivations of the Neural Network Research	19
3.4	History and Background	20
3.4.1	A Brief History of Neural Networks	20
3.4.2	The Perceptron	21
3.4.3	The Multilayer Perceptron	22
3.4.4	The Hopfield Net	23
3.4.5	The Hamming Net	25
3.5	Neural Networks for Pattern Recognition	27
II	Methodology	29
4	Pattern Designation	30
4.1	What is a Pattern?	30
4.2	What is Pattern Designation?	31
4.3	Domain Discretization	33
4.4	Partitioning Parameter Space	37

4.4.1	Algorithm 4.1: Top down dichotomy	38
4.4.2	Discussion and Analysis of Algorithm 4.1	39
4.5	Sample Size Determination	41
4.5.1	Sampling Schemes	41
4.5.2	The Efficiency of Sampling	42
4.5.3	Estimating the Sampling Error of Systematic Sampling in One Dimension Space	44
4.5.4	Estimating the Sampling Error of Stratified Random Sam- pling with One Sample Point Per Stratum	48
4.6	Summary	53
5	Feature Extraction	55
5.1	Introduction	55
5.2	Objective	56
5.3	Karhunen-Loève Expansion	57
5.4	Fisher's Discriminant Method	59
5.5	An Improvement on the Fisher's Discriminant Method	63
5.6	Some Other Analytical Techniques	64
5.7	Summary	66
6	Classification	67
6.1	Introduction	67
6.2	Conventional Classifiers	67
6.2.1	The Classifiers of Statistical Pattern Recognition	68
6.2.2	The Classifiers of Syntactic Pattern Recognition	69
6.3	Neural Network Classifiers	69
6.3.1	The Neural Network Classifier Based on the Hamming Net	70
6.3.2	Modified LVQ	72
6.4	Example	76
6.5	Discussion	82
6.5.1	Training with Noise	82
6.5.2	Null Class Classification	85
6.6	Conclusions	86
7	Performance Evaluation	88
7.1	Testing Set Generation	88
7.2	Estimating the Confidence Level of Classification	89
7.3	Class Region Growing	90
7.4	Class Region Merging	93
7.5	Summary	93

III Applications

8	Case Study I: Characterization of a Nonlinear Torsional Spring	95
8.1	Introduction	95
8.2	The Simple Articulated Joint System	95
8.2.1	The System	95
8.2.2	The Objective	96
8.2.3	Classification Results	98
8.3	The Need for the Pattern Recognition Approach	106
8.3.1	Experiment Design	107
8.3.2	Simulation Results	107
8.3.3	Discussion	108
9	Space Station Freedom Model	111
9.1	Introduction	111
9.2	The Space Station Freedom Model	111
9.2.1	The System	111
9.2.2	Objective	112
9.2.3	Experiment Design	112
9.2.4	Experiment Data	112
9.2.5	Classification Results	115
9.3	Conclusion	119
10	Aircraft System	122
10.1	Introduction	122
10.2	The Aircraft System	122
10.2.1	Aircraft Reference Coordinate Systems	122
10.2.2	Control Surfaces	124
10.2.3	Equations of Motion	126
10.3	F-15 Flight Simulator	127
10.4	The Objective	131
10.5	Experiment Design	131
10.6	Classification Results	134
10.7	Experiment Results on the SRV Flight Data	136
10.7.1	Flight Data	136
10.7.2	Objective	137
10.7.3	Experiment Design	137
10.7.4	Experiment Results	139
10.8	Discussion and Conclusions	140
11	Conclusions	142
11.1	Summary	142
11.1.1	Pattern Designation	143
11.1.2	Feature Selection	143
11.1.3	Classifier Implementation	144

11.1.4	Performance Evaluation	144
11.1.5	On-line Identification	145
11.2	Limitations	146
11.3	Implications	146
11.4	Applications	146
11.5	Contributions	147
11.6	Suggestions for Future Research	148
11.6.1	Syntactic Pattern Recognition Approach	148
11.6.2	Reduction of the Sample Size	149
11.6.3	Other Applications	149
A	Splitting Dimension Analysis	150
B	Generalized Eigenvalue Problem	152
	References	154

LIST OF FIGURES

2.1	A model structure.	9
2.2	A general configuration of parameter estimation procedure.	12
2.3	A function and its identified functional region.	15
2.4	(a) An optimization search fails to locate the true solution. (b) The PR approach identifies a reduced parameter space which contains the true solution.	16
3.1	(a) A simple processing unit, where $y = f(\sum_{i=1}^n w_i x_i - \theta)$. (b) Three types of nonlinearities.	18
3.2	A Hamming net.	26
4.1	(a) A possible curve of the unknown function $f(x)$. (b) A piecewise linear approximate.	36
5.1	A neural network implementation with perceptrons of the feature extractor.	61
6.1	A neural network classifier.	71
6.2	Three pattern classes and the initial locations of exemplars for the three classes using MLVQ.	76
6.3	The final locations of the exemplars using MLVQ.	77
6.4	The number of misclassifications in the training history using LVQ and MLVQ with 3, 5, and 15 exemplars per class.	78
6.5	Intensive learning vs random learning.	80
6.6	The initial locations of the exemplars with biased distribution using ADSM.	81
6.7	The final locations of the exemplars using ADSM.	81
6.8	Misclassification rates (%) of MLVQ against noise levels under training with noise scheme.	84
6.9	Misclassification rates (%) against the number of noisy patterns per original training pattern used in training with noise scheme.	85
7.1	The mapping from the parameter space to the pattern vector space.	91
7.2	The parameter space overlap between class i and class j	92
8.1	The articulated joint system of two rigid bars connected by a torsional spring.	96
8.2	(a) A simulation result of system response at $k(\theta) = 29.61$. (b) A plot of $\omega(t)$	97

8.3	The piecewise linear function approximates the spring torque function.	99
8.4	(a) The operational region of the spring torque function. (b) the new parameter space.	100
8.5	The partitioned parameter space, where the label on each cubic corresponds to its class identity.	100
8.6	The binary tree of the partition process.	101
8.7	The classes of the spring torque functions correspond to the regions in the partitioned parameter space.	102
8.8	A bi-linear approximation of the spring torque function.	103
8.9	The partitioned parameter space.	104
8.10	The true spring torque function (solid line) and the functional region identified via the PR approach	108
8.11	The spring torque function obtained via the DSM (solid line) and the functional region identified via the PR approach	109
8.12	The spring torque function obtained via the DSM (solid line) and the functional region with which it is classified via the PR approach	110
9.1	The Space Station Freedom model.	113
9.2	A system acceleration response in the time domain.	114
9.3	The possible region of the unknown damping torque function.	116
9.4	The pattern classes.	117
9.5	A system response with and without noise.	118
9.6	A system response with 10% and 20% noise at $MNL = 0.0001$	120
10.1	The aircraft body-axis system and control surfaces.	123
10.2	Vehicle-carried vertical axis system.	123
10.3	Rotation of axes through Euler angles.	125
10.4	Flow angles and relationship of body and wind axes.	126
10.5	The flow chart of the F-15 flight simulator.	129
10.6	(a) The maneuver of elevator. (b) The response of normal acceleration. (c) The response of the angle of attack α	132
10.7	A possible functional region of the lift coefficient which is a surface between the upper surface and the lower surface.	134
10.8	The response of normal acceleration in time domain corrupted by Gaussian noise with 0.0414 standard deviation and 0 mean.	135
10.9	The functional region containing the true lift coefficient.	135
10.10	(a) The maneuver of elevator. (b) The response of normal acceleration. (c) The response of the angle of attack α	138

LIST OF TABLES

3.1	A taxonomy of neural network for pattern recognition.	28
6.1	Misclassification rates (%) on the testing set using different methods.	79
8.1	System parameters.	96
8.2	Different levels of Gaussian noise and their corresponding SNR. . .	101
8.3	Classification results in terms of error rates (%) for the training set and testing set contaminated by different levels of Gaussian noise. .	103
8.4	Classification results in terms of error rates (%) for the training set and testing set corrupted by different levels of Gaussian noise. . . .	105
9.1	Classification results with noise of fixed standard deviations.	116
9.2	Classification results with noise proportional to signal's amplitude .	119
10.1	Classification results for the training set with Gaussian noise.	135
10.2	Classification results for the training set of the second level partition with various levels of Gaussian noise.	136
10.3	The scaling relationships between the F-15 and the SRV.	137
10.4	Percentage of correct classification results for the training set and the SRV flight data of the second level partition with various levels of Gaussian noise.	140

GLOSSARY

Alpha Gimbal A one-degree-of-freedom joint connecting an extraneous body to the central body of the Space Station Freedom model.

Class A set of patterns clustered with some common properties. Also called **Pattern Class**.

Dynamic System A system whose current output value depends not only on the current inputs but also on their earlier values.

Feature A variable or characteristic of patterns utilized by a classifier to separate patterns into classes.

Feature Space A linear space constructed by the feature vectors.

Feature Vector A vector representation of feature components.

Functional Region An open region in a function space to encapsulate functions in the function space.

Function Space A linear space where the domain and range of a given function are represented.

Input The external stimulus to a system.

Lift Coefficient An important parameter associated with the combined aerodynamic force on an aircraft normal to the aircraft velocity vector.

Model A description that characterizes the properties of a real world system. A model may come in a linguistic or mathematical expression. It is referred to be a mathematical model in this dissertation.

Model Class A set of model with some common properties. It is interchangeable with **Pattern Class** in the dissertation.

Model Set A collection of models.

Model Structure A parametric model with undetermined parameters.

Output The observable signal generated by a system.

Parameter Space A linear space defined by the coefficients of a model structure in order to represent a parametric model set.

Pattern A conceptual representation of an object in ordinary pattern recognition applications. In this dissertation, a pattern is referred to a mathematical model characterizing a real world system or subsystems in a complex system being modeled.

Pattern Class A set of patterns clustered with some common features.

Pattern Vector A physical representation of a pattern in a form suitable for machine processing.

Pattern Vector Space A linear space constructed by the pattern vectors.

System An object that is affected by external excitations and produces observable signals.

System Response The output of a system or model for a given input.

ACKNOWLEDGMENTS

This dissertation would not have been possible without the help of many. I would like to thank you all. In particular, I thank:

Professor Walter Karplus, my advisor, for his support, encouragement, guidance and for his ideas that started this work;

Professors Jack Carlyle, Jacques Vidal, Nhan Levan, and Edward DeLand, for valuable comments and for serving my doctoral committee;

Dr. Ray Gluck, for his inspiration on the problem of nonlinear structural parameters realization and for presenting me the simple articulated joint system which was then used as a testbed for the proposed methodology and accelerated this research;

Dr. Richard Maine, for the SRV flight data;

Bob Tisdale, for invaluable discussion, installing the Space Station Freedom and F-15 simulators as well as for his constant curiosity about my work which pushed me through the completion of this dissertation;

Eric Shank, for help in aerodynamics;

Chun-Houh Chen, for help in statistics;

June Myers, for administrative assistance;

Dr. Boris Kogan, Brian Billett, Masahiro Kayama, Dr. Xinming Lin, and Mike Stiber for making the research environment stimulating and exciting; and

The Chinese student society in Computer Science Department, Dr. Li-Wen Chen, Ming-Yung Horng, James Liu, Wen-Toh Liao, Jonathan Lu, Chien-Chung Shen, Dr. Yea-Li Sun, Yih-Kuen Tsay, and Ping-Hann Wang ... etc., for providing a friendly environment and joyful lunch time discussion which made life easier when research went nowhere.

In addition, I would like to thank my brothers, Han-Cheng and Han-Ping, for

taking care of my father so that I can concentrate on my research. This dissertation is dedicated to my father, Chin-Rong Dai, and to the memory of my mother, Siue-Lien Hwang, for their everlasting love and support. Finally, I thank my wife, Woan-Ling, for her encouragement, understanding and being patient with my moods.

This research is supported in part by TRW under MICRO program and by NASA grant NCC 2-374.

VITA

January 26, 1961	Born, I-Lan, Taiwan
1983	B.S., Computer Science National Taiwan University Taipei, Taiwan
1985 – 1986	Research Assistant Institute of Information Science, Academia Sinica Taipei, Taiwan
1988	M.S., Computer Science University of California, Los Angeles
1989 – present	Post Graduate Research Engineer Computer Science Department University of California, Los Angeles

PUBLICATIONS

Rafael Gluck, Han-Sen Dai, and Walter J. Karplus. On-orbit nonlinear structural parameters realization via artificial neural network. To appear in *The AIAA/ASME/ASCE/AHS/ASC 33rd Structures, Structural Dynamics and Materials Conference*, April 1992, Dallas, Texas.

ABSTRACT OF THE DISSERTATION

System Identification Using Neural Networks

by

Han-Sen Dai

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 1992

Professor Walter J. Karplus, Chair

The research described in this dissertation is devoted to the development of new and improved methods for the modeling and simulation of physical dynamic systems. The focus is the characterization in identification of the nonlinear parameters that appear in the governing differential equations. In the approach presented here, parameter characterization is treated as a pattern recognition problem. Noisy data obtained from observation of the dynamic system constitute the input to the pattern recognition system which includes a feature extractor and a pattern classifier.

The probable candidate functions for the unknown parameters are designated as "patterns." Computer simulations are conducted over a variety of candidate mathematical models of the system with selected functions for the parameters. A function space encompassing all possible patterns is automatically partitioned into pattern classes in a top down dichotomous manner, which performs partitions

based on the variation of the simulated system responses and introduces coherent property within any pattern class. An adaptive artificial neural network serves as a pattern classifier. The actual system response is classified with the model class of the most similar system response. The identified model class contains a model set that is a most likely representative of the system.

The proposed methodology is applied to characterize the important components of real world systems, in particular, the Space Station Freedom and high performance aircraft. The results are encouraging, showing a high percentage of correct classifications in a noisy environment. When optimization search techniques are conducted to search for the best model, the identified model class which is a reduced function space provides useful information for choosing an appropriate starting model.

CHAPTER 1

Introduction

System identification deals with the problem of constructing a mathematical model of a physical dynamic system based on the observation of system inputs and outputs. The significance of this problem can be addressed in several application areas: simulation, control, and prediction. All of the applications need accurate mathematical models in order to obtain desired results and performance. In simulation applications, accurate models are necessary to achieve high fidelities of the real world systems. Simulations also help scientists and engineers to understand the system being analyzed in a great deal because some experiments on the system are very difficult to conduct in the real world, but easy to implement by simulations. If the model is not accurate enough to demonstrate the characteristics under study, simulation results just lead to an incorrect conclusion. In control applications, mathematical models provide tools to analyze a system to be controlled and to design control strategies for the system.

Models can be obtained from physical reasoning or by analyzing observed data from the system. Often both approaches are required to build a model for a complex physical system. Physical reasoning, also known as deduction or synthesis approach, is out of the scope of the dissertation. On constructing a mathematical model, an appropriate mathematical structure which presents a set of models by varying unknown parameters should be obtained first, and then the best values of the unknown parameters of this mathematical structure are estimated. These are the two major steps of system identification, namely system characterization and

parameter estimation, respectively.

This study emphasizes the problem of system characterization. This problem is often attacked by using a priori knowledge or engineering insight. The approach employed in this study is based on the pattern recognition technique. Computer simulations are conducted over a set of candidate models. The simulated system responses are organized into clusters which correspond to pattern classes and are associated with their models. When the actual system response is presented, the real world system is classified with the model class which associates the most similar system response.

Recent advances in the field of massively parallel artificial neural networks have made it possible to achieve human-like problem solving performance, especially in pattern recognition. It is plausible to consider the application of the new pattern classifiers and associative memory based on neural network models in the system characterization problem.

1.1 Objectives

Conventional system identification approaches are based on optimization techniques to minimize a criterion function which specifies the goodness of fit between the model and the true system [Bek70]. These approaches begin with a starting model and iteratively refine the model until a local minimum of the criterion function is reached. For example, the nonlinear gradient decent search is a well known and widely used technique in system identification. However, the optimization techniques may take many iterations and fail in several situations such as low quality observations, incorrect model structure, or bad initial values of undetermined coefficients. Instead of obtaining a global minimum, these techniques may end up with a local minimum which makes the identification result of no use.

This research investigates a complement to conventional approaches. The results obtained from the proposed methodology can provide useful information for the optimization approaches with a better starting model in order to reduce search steps as well as to avoid local minima hopefully.

The main objectives of the research are:

1. Development of a methodology using neural networks for nonlinear system characterization, including:
 - (a) Formulation of the characterization of dynamic systems as a pattern recognition problem.
 - (b) Design and implementation of a feature extractor and a pattern classifier using neural networks.
 - (c) Evaluation of the reliability of classification results.
2. Application of the methodology to two real systems:
 - (a) Identification of the nonlinear characteristics of the damping torques at the Alpha gimbals of the Space Station Freedom model.
 - (b) Characterization of the nonlinear lift coefficient of the high performance aircraft like the F-15.

1.2 Methodology

In this research, system characterization is formulated as a pattern recognition problem. First, a pattern designation technique is employed to generate patterns and pattern classes corresponding to candidate mathematical models and model sets, which are likely representatives of the real world system. This includes approximation of unknown nonlinear functions of the system, construction of a pa-

parameter space representing candidate models, and partition of the parameter space into pattern classes.

Secondly, a feature extractor and a pattern classifier are implemented with neural networks in order to enhance classification performance and to achieve the capability of rapid recognition.

Finally, performance evaluation is carried out to provide the confidence information for characterization results. This includes the estimations of misclassification rates and the effect of noise on classification.

1.3 Organization of Dissertation

The dissertation is divided into three parts: background, methodology, and applications. Part I contains Chapter 2 and Chapter 3 which give brief background on the system identification problem and neural networks as well as their relation to this work. Part II, from Chapter 4 to 7, describes the proposed methodology for system characterization with neural networks. Part III presents the applications of the methodology to three case studies, including Chapter 8 to 10.

Chapter 2 describes the concepts and notions of the system identification problem. This is followed with a discussion of the conventional optimization approach and the pattern recognition approach to attacking this problem. Then a problem statement is presented to be the scope of this work.

Chapter 3 deals with neural network problem solving models. First, the concepts and the attractive features of neural networks are described. Second, a brief history of neural network is given. Furthermore, various models of neural networks are discussed. Finally, the neural network models for pattern recognition applications and their taxonomy are presented.

Chapter 4 discusses the definition of a pattern in the system identification prob-

lem domain. The construction of a parameter space for representing the candidate models is given. Pattern class designation is performed automatically via a top down dichotomy algorithm to partition the parameter space into pattern classes.

Chapter 5 presents the Fisher's discriminant functions as the feature extraction scheme. Frequency domain analysis and exponential components analysis techniques are discussed for extracting nonlinear features.

Chapter 6 introduces an adaptive pattern classifier with neural network implementation. Besides, the performance of this classifier is investigated and discussed.

Chapter 7 is devoted to the estimation of classification performance. A region growing technique is also presented to make the classification result more reliable and useful.

Chapter 8 reports on the application of the proposed methodology to characterize a torsional spring at an articulated joint of a simple two bar system. Plausible results are presented. Particular attention is focused on a comparison of identification results of an optimization approach and the proposed methodology.

Chapter 9 describes the application to identify the characteristics of the damping torques at the Alpha gimbals of the Space Station Freedom model.

Chapter 10 presents the application to characterize the lift coefficient of the F-15 aircraft and the 3/8-scale F-15 airplane model.

Chapter 11 concludes this work and recommends future research.

Part I

Background

CHAPTER 2

Problem Formulation

2.1 Introduction

This chapter opens with a general description of the system identification problem followed by a discussion of its four subproblems. Then the optimization approach and pattern recognition approach are described. Finally, the problem statement of this study is given.

2.2 System Identification

A system is a physical process that is affected by external stimuli and generates observable signals. The external stimuli are called inputs or excitations to the system. The observable signals produced by the system are called outputs or responses. A system is said to be dynamic if its current output depends on the current inputs and their earlier values. Therefore, dynamic systems have memories about their earlier status or possess system “states.”

System identification is concerned with the problem of building mathematical models of real world dynamic systems based on the observations of inputs and outputs of the systems. Conceptually, a system is a function which maps inputs to outputs, and an identification method deals with a mapping from the space of observations to a space of models. This is an inverse problem. Its solution may not be unique.

A model of a system is a description that characterizes its properties in order

to fulfill a certain purpose. Models may come in various shapes (e.g., descriptive or linguistic models). Some models involve mathematical formalizations, while others do not. For applications in the engineering field, it is necessary to use models that describe the systems in terms of mathematical expressions. We shall refer to such models as mathematical models. The accuracy of a model would certainly depend on its purpose. For instance, models for aerospace applications need to be very precise.

Definition 2.1 *A model set \mathcal{M}^* is a collection of models, where the search for a suitable model is conducted over.*

To put this definition in mathematical notation we have

$$\mathcal{M}^* = \{\mathcal{M}_\alpha | \alpha \in \mathcal{A}\} \quad (2.1)$$

Although the models are enumerated with an index α covering an index set \mathcal{A} , a model set of interest is often uncountable. Since we need to perform a search over the model set for the best model, one approach is to parameterize the set and conduct the search over the parameter set.

Definition 2.2 *A model structure \mathcal{M} is a mapping from an open subset Θ of \mathbf{R}^d to a model set \mathcal{M}^* .*

That is

$$\mathcal{M} : \Theta \ni \theta \longrightarrow \mathcal{M}(\theta) \in \mathcal{M}^*, \quad (2.2)$$

where θ is the parameter and Θ defines the parameter space. Thus the model structure parameterizes the model set with a parameter set Θ . Also $\mathcal{M}(\theta)$ denotes the particular model corresponding to θ .

Consider a system with an input signal $u(t)$ and an output signal $y(t)$. The observed signal $z(t)$ is the output signal disturbed by a measurement noise $\xi(t)$.

The system is said to be *causal* if the output at a certain time depends on the input up to that time only. Causality is a condition necessary for identifying the system from the input and output observations. If this is a dynamic system, a set of auxiliary variables $x(t)$ is used to represent the internal states of the dynamic system. Figure 2.1 shows a model structure for the system with a parameter set Θ .

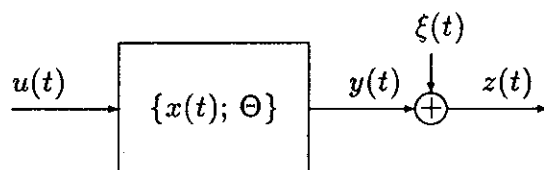


Figure 2.1: A model structure.

In general, the construction of a model from observed data involves four steps:

1. experiment design,
2. system characterization,
3. parameter estimation, and
4. model validation.

The first step is to design the identification experiment so that the observations become suitably informative. The design inputs should be able to excite the unknown components of the dynamic system being modeled and must be operationally executable in the real world. Secondly, system characterization determines the mathematical structure which characterizes the system. That is to select \mathcal{M} and Θ . This step is often accomplished via a priori knowledge and engineering insight. Parameters of the selected mathematical structure are estimated in the third step. In

general, optimization search techniques are conducted over Θ to identify a parameter θ for the best model. Once a model is identified, it remains to test whether the model is good enough in the final step. The whole identification procedure is performed iteratively. In other words, the identified model can be revised or refined by repeating the procedure if it is rejected in the model validation step.

Depending on the a priori knowledge of the system under consideration, the identification problems can be classified into three types: white box, gray box, and black box. A problem is said to be a “white box” if the mathematical structure describing the system is explicitly known. System characterization is well defined in this case. For instance, an electrical circuit may belong to this category. On the other hand, a system with little or no a priori knowledge about its mathematical model is considered as “black box” type. Social and political systems are examples of this type. Between these two extreme types, there are systems for which the general form of the mathematical structure is known, but an explicit one is not known. These systems are referred to as “gray box” type, such as air pollution and aircraft control. The problems of interest in identification are usually the gray box type.

There are two types of parameterized model sets corresponding to the black box and gray box problems:

1. *Black box model structure:* The basic idea is to obtain flexible model sets without looking into the system internal structure. This type of model structures does not provide any insight into the system components.
2. *Mechanistic model structure:* This approach constructs model structures by the deduction of the “law of nature.” This type of model structures comes with adjustable parameters which provide physical interpretations.

2.3 Related Work

Parameter estimation has been well described in the literature [AE71, Bek70, Lju87], but relative little work has been done on system characterization [Kar72, AK82, SK86, SH74, Sim75]. This is because a model structure for parameter estimation is usually given by experienced experts with engineering insight into the real world system. Given a model structure, the parameter estimation problem is often performed via optimization techniques [Bek70, PFTV88]. Although the system characterization problem is difficult to formalize into algorithms and to automate, the pattern recognition approach has been successfully applied to attack this problem [SH74, SK86, Sim75].

2.3.1 Optimization Approach

The conventional parameter estimation approach consists of three steps:

1. Select a model structure that is likely to represent the real world system.
2. Define a criterion function which measures the goodness of fit of the model responses to the real world system responses.
3. Specify a computational procedure for adjusting the parameters of the selected model in order to minimize the criterion function.

Figure 2.2 shows a general configuration of parameter estimation procedure.

Unless the parameters are linear in the criterion function, the computational procedure to minimize the criterion function is usually carried out by iterative optimization techniques, which require good initial guesses of the parameter values for convergence to the correct solution and which may take many iterations to converge. Its performance also strongly depends on the assumption of the underlying

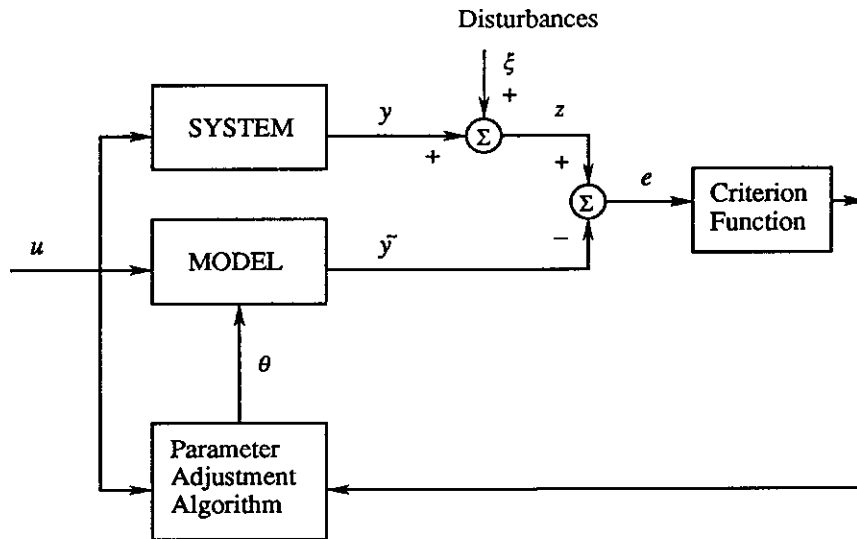


Figure 2.2: A general configuration of parameter estimation procedure.

model structure. If the model structure is not good enough, the solution obtained is of little use. Furthermore, low quality of observations can prevent convergence of the optimization procedure.

2.3.2 Pattern Recognition Approach

Karplus [Kar72] first proposed an approach for identification and simulation of systems using the pattern recognition technique. Saridis and Hofstadter [SH74] applied this technique to characterize nonlinear systems by the correlation and autocovariance between system inputs and outputs. Simundich [Sim75] accomplished system characterization of parabolic distributed parameter systems via this approach. Shibata and Karplus [SK86] carried out the identification of water pollution source using spectrum analysis as features for pattern recognition. The pollution source was specified by a time function and a space function. The time function was first identified by the pattern recognition approach utilizing the coherence function of power spectra. Secondly, the least squares method was applied

to determine the space function.

The pattern recognition (PR) approach for characterization includes four steps:

1. *pattern designation*: A variety of candidate models are designated to be patterns which are clustered into classes.
2. *feature extraction*: Feature vectors are extracted from system responses to represent the unique properties of the real world system and the candidate models.
3. *pattern classification*: A pattern classifier is trained to classify the feature vector of the real world system into one of the classes of the candidate models.
4. *performance evaluation*: The performance of the pattern recognition system in modeling is evaluated. And the model obtained is validated.

2.4 Problem Statement

On constructing a model for a real world system, deduction rules and a priori knowledge help to derive the model up to some extent. The remaining unknown part of the model has to be determined from the observations of system's input and output data. This research deals with the mechanistic model structures containing some unknown parameters which may be functions of system states and inputs. Suppose that there exists very limited prior knowledge about the behavior of these parameters or unknown functions. The effort of this research attempts to identify the general shapes of the curves or surfaces of these unknown functions from the observed data of the real world system in a noisy environment. Besides, these data may be sampled both in time and space.

As a matter of fact, the general shape of a curve or surface corresponds to a *functional region* in the space constructed by the unknown function and its domain

space. For instance, if the domain and range of the unknown function $f(x)$ are both one dimensional space, a functional region that characterizes the curve shape of $f(x)$ may be a band in the two dimensional space constructed by $f(x)$ and x . Figure 2.3 shows a true function of the real world system classified with one of three candidate functional regions.

In this research, the proposed methodology is based on the principle of the pattern recognition approach. Instead of fitting a model to the detail of every observed data point, the PR approach utilizes the characteristics of candidate models and the system to be modeled. The features of the system are extracted from the observed data for pattern matching. There are three problems involved:

1. designation of these possible general shapes for pattern classes.
2. design and implementation of the feature extractor that extracts the invariant features of each pattern class.
3. design and implementation of a classification scheme.

Once a functional region of the unknown function is identified, a better mathematical structure close to the identified shape can be constructed for further refinement to locate the true function. More appropriate initial values of undetermined parameters of the new mathematical structure are also imposed in the functional region because it restricts the ranges of the parameters.

2.5 Why is Pattern Recognition Approach Important to System Identification Problem?

The optimization approach for parameter estimation needs a mathematical form of $f(x)$ which defines a parameter space for searching the numerical values of the parameters. Figure 2.4(a) presents a scenario of an optimization search

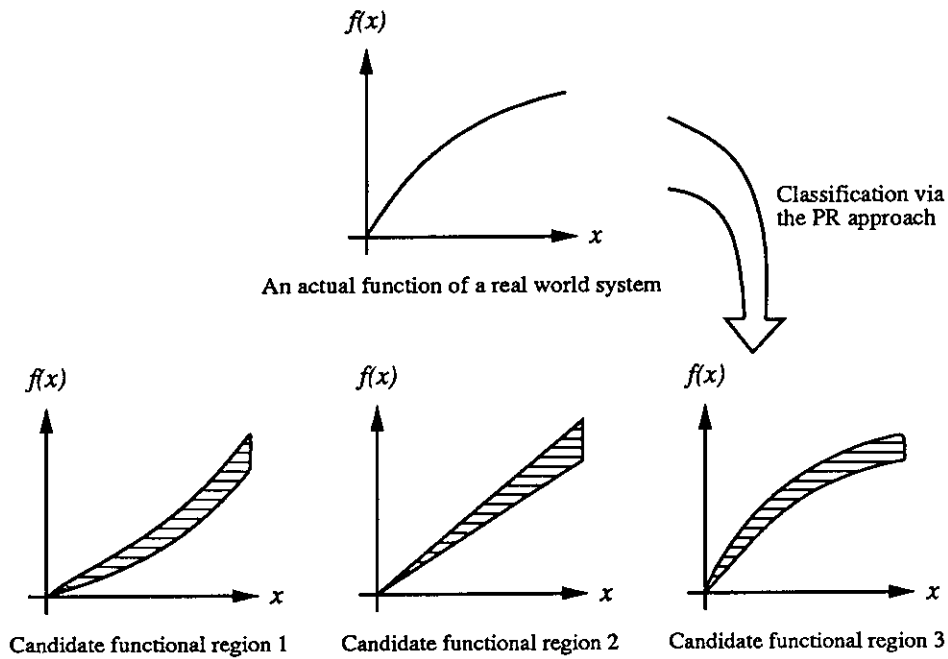


Figure 2.3: A function and its identified functional region.

conducted over the parameter space. Due to the aforementioned reasons, the optimization search falls short.

On the other hand, the PR approach classifies the observations of the real world system with one of the predetermined model classes that characterize the function $f(x)$. Having determined the region which $f(x)$ falls, we have narrowed the possible range of $f(x)$ instead of locating the exact solution of the unknown function in a vast space. Figure 2.4(b) shows that the partitioned parameter space with regions corresponding to classes and a parameter space reduction process using the PR approach. A better starting point which is close to the solution is now available. In addition, a more precise mathematical structure can be constructed through the reduced parameter space. Thus, the result derived from the PR approach can provide more information for the optimization approach in selecting an appropriate mathematical structure and initial values of parameters, especially when low

quality observations are presented or the system is deficient in prior knowledge.

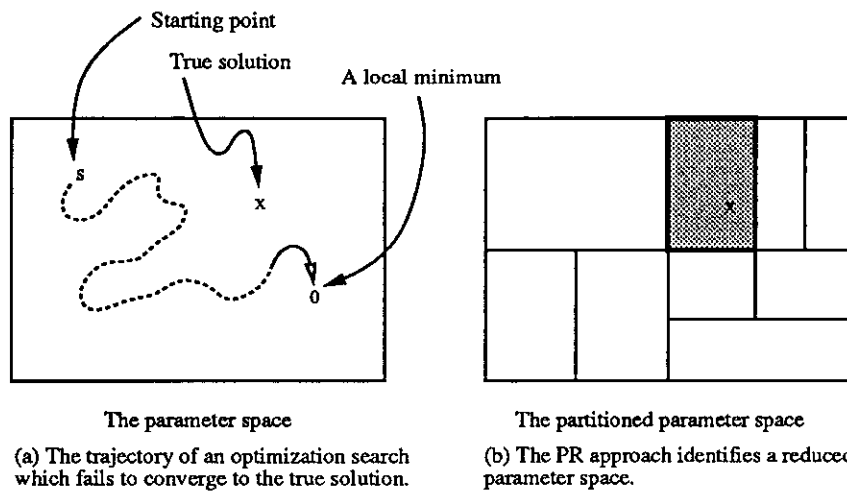


Figure 2.4: (a) An optimization search fails to locate the true solution. (b) The PR approach identifies a reduced parameter space which contains the true solution.

CHAPTER 3

Neural Networks

3.1 Introduction

Artificial neural networks have become very attractive computational tools. It is an area of intensive activity and growing interest in practical applications because of the new network architectures and training algorithms, analog VLSI and optical implementations, and the belief that massive parallelism is essential to achieve human-like performance. In the beginning of this chapter, the attractive features of neural network computational model are described. Second, we present the motivation of this prosperous research area. Third, a brief history of neural networks is given. Various models are also discussed.

3.2 Definition and Attributes

Artificial neural networks were originally employed in developing mathematical models of biological neurons. The synapses are characterized by connection weights in the artificial neural network models, while the frequency modulation of nerve impulses is modeled by an amplitude modulation of nonlinear activation functions. Recently, it was realized that human-like performance in speech and image recognition requires enormous amounts of computation. Neural networks provide a technique for obtaining the required computation using a large number of simple processing units operating in massively parallel.

Because of various neural network models, it is not necessary to give a unified

definition. However, the attributes of neural networks may justify their definition. These attractive attributes are listed as follows:

- *Simple processing unit:* Each processing unit (neuron or node) simply computes the weighted sum of its inputs and then applies a nonlinear activation function to generate its output. Figure 3.1 (a) shows a single processing unit and its computational function. In general, there are three types of nonlinear activation functions, namely hard limiter, threshold logic, and sigmoid, as shown in Figure 3.1 (b).

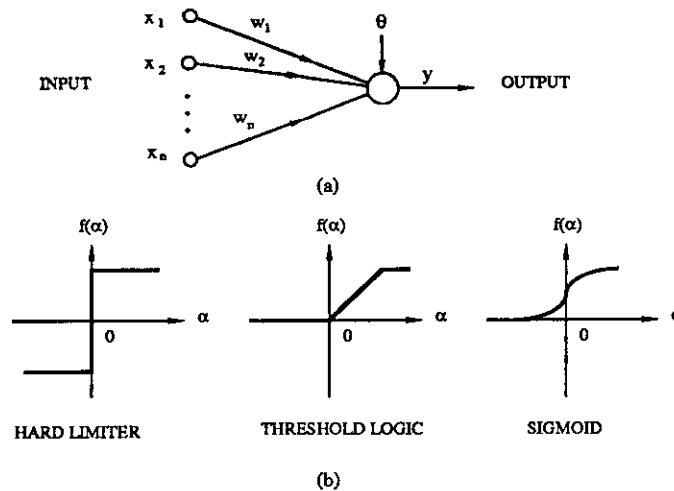


Figure 3.1: (a) A simple processing unit, where $y = f(\sum_{i=1}^n w_i x_i - \theta)$. (b) Three types of nonlinearities.

- *High connectivity:* Similar to biological nervous systems, neural network models attempt to achieve good performance via dense interconnections of numerous simple processing units.
- *Massively parallel:* The network with large number of processing units operates in massively parallel.
- *Fault tolerance:* In contrast to conventional computers, the malfunction of

several processing units or connection links might not affect the system performance too much.

3.3 Motivations of the Neural Network Research

There are several motivations for this research area:

- *Biological understanding:* Neural networks may provide models for studying biological nervous systems. The work combining laboratory experiments and network models can verify the assumptions of network theory and improve models. Although network models used today are very simple in comparison to biological systems, the eagerness to understand brain function always motivates this research area.
- *Mind understanding:* Neural network models have already had an important influence on cognitive science. A new class of theories of mind is emerging because of the network models which provide a new approach to psychological modeling.
- *Problem solving:* For cognitive applications, the practical systems are of essential interest, such as speech and image recognition, vision, and sensory motor control. Neural network models have been studied in hope of supporting an efficient architecture and achieving human-like performance in these fields that are far beyond the performance of the current best artificial systems. In addition, other applications, such as optimization, control, and prediction, also stimulate this research area.

3.4 History and Background

3.4.1 A Brief History of Neural Networks

In 1943, McCulloch and Pitts [MP43] invented the threshold logic model which was the first effective mathematical formalization of the operation of a neuron. In 1949, Hebb [Heb49] introduced the famous Hebbian Learning Rule which specifies a self-organizing model of the synapses. Nowadays most of the learning rules are adaptations of this early principle.

Rosenblatt [Ros58, Ros62] proposed a learning machine with Hebbian learning rule called the *perceptron*. He also proved the remarkable *Perceptron Convergence Theorem* about the perceptron learning rule [Ros62]. The invention of the perceptron was a major advance toward a formal model of a neural network. Meanwhile, Widrow and Hoff [WH60] introduced an adaptive pattern classification machine called *adaline*. An “adaline” was a threshold logic unit with variable connection strengths. The connection strengths were adjusted based on the *error* between what the output was supposed to be and what the adaline computed. The search over this error surface was in a sense of gradient descent to minimize the total error. The best known supervised learning algorithm today, “back propagation” [RM86], is a generalization of the simple Widrow-Hoff rule.

In 1969, Minsky and Papert [MP69] published a book “Perceptrons” which shocked the foundations of this field and discouraged research for many years. However, this book proved some formal results about perceptrons and exposed the inherent problems of perceptrons.

During the dark age of this research field, several researchers still immersed themselves in hard work. Later, their efforts did flourish, such as

- Kohonen’s self organizing feature maps and Learning Vector Quantizer (LVQ)

[Koh84],

- Carpenter and Grossberg's Adaptive Resonance Theory (ART) [CG86],
- Fukushima's Neocognitron [FMI83],
- the Hopfield net [Hop82, Hop84],
- the multilayer perceptron and the back propagation learning algorithm [RM86], and
- Sejnowski and Rosenberg's NETtalk [SR86], etc.

The breakthrough of [RM86] was to modify the design of the perceptron in order to solve the learning problem in multilayer networks. The solution was to replace the threshold activation function of the perceptron by a smooth sigmoid function. Since the sigmoid function is continuous and differentiable, it is possible to derive a gradient descent learning algorithm. NETtalk is an application of multilayer perceptron to speech. The impressive demonstration of NETtalk shows the promise of neural networks.

3.4.2 The Perceptron

Minsky and Papert [MP69] gave the definition of perceptrons as follows: "*A perceptron is a device capable of computing all predicates which are linear in some given set Φ of partial predicates.*"

In this definition, Φ is a collection $\{\phi\}$ of predicates with values 0 or 1. Let $\{\alpha\}$ be a set of number indexed by Φ and θ be some number for a threshold. Mathematically, the perceptron computes

$$P(\{\alpha\}, \Phi, \theta) = \begin{cases} 1 & \text{if } \sum_{\phi \in \Phi} \phi \alpha_{\phi} > \theta \\ 0 & \text{otherwise} \end{cases} \quad (3.1)$$

There are various types of perceptrons given in the book “Perceptrons” [MP69]. The most common perceptron seen today is the one shown in Figure 3.1 (a). In this case, α_ϕ corresponds to a connection weight w_i and Φ is the input.

The learning rule operates repeatedly by adjusting $\{\alpha\}$ and testing the perceptron to improve the performance. It turns out that the perceptron can learn some functions, but the range of functions realizable by single layer perceptrons is so limited. The exclusive-or (XOR) problem is a famous example.

3.4.3 The Multilayer Perceptron

Perceptrons can be organized into layers with a number of perceptrons in each layer. In the layered network, the outputs of one layer are the inputs to the subsequent layer. With these cascade perceptron layers, the perceptron learning rule does not apply to the multilayer structure because of the special nonlinearity of the threshold logic unit. This learning problem had made the multilayer perceptron impractical to apply for many years.

In the early 1980’s, a breakthrough was made by replacing the threshold logic unit with a smooth sigmoid function. As the sigmoid function is differentiable, this modification led to a gradient descent learning rule called *Generalized Delta Learning Rule* (back propagation algorithm) [RM86]. Back propagation is a supervised learning algorithm which need a “teacher.” It applies a gradient search technique to minimize the mean square error between the desired and the actual network output by adjusting the connection weights. The error terms are propagated back from the nodes in the output layer to nodes in lower layers, so the connection weights can be adjusted iteratively and sequentially in the layered error propagation order.

Theoretically, a two-layer perceptron is sufficient to construct convex decision

regions of pattern classes. With a three-layer structure, the decision region of a pattern class can be arbitrary as long as there are enough internal perceptrons [Lip87]. The XOR problem is easily solved. However, the major problem with the generalized delta learning rule is that the learning is not only very slow, but the gradient descent search method tends to get stuck at local minima.

3.4.4 The Hopfield Net

The Hopfield net [Hop82, Hop84] consists of a fully interconnected network of many simple processing units operating in a massively parallel fashion. The processing units, or nodes, which are connected by links with variable weights, are able to compute a linear summation of scalar input values along with a nonlinear activation. The node is characterized by an internal threshold θ and by nonlinear activation functions of the types presented in Figure 3.1. In the following discussion, we only deal with the hard limiter activation function.

A Hopfield net with n nodes can be specified by an $n \times n$ connection matrix W and a threshold vector $\vec{\theta}$. We have $W = [w_{ij}]$ and $\vec{\theta} = [\theta_1, \theta_2, \dots, \theta_n]^T$, where w_{ij} is the connection weight from node j to node i , and θ_i is the threshold of node i . Each node is capable of outputting one of two values: either +1 (on) or -1 (off). The state of the system as a whole at time t is described by a state vector $V(t) = [v_1(t), \dots, v_n(t)]^T$. The operation of node i is defined as:

$$v_i(t+1) = \mathbf{sign}(u_i(t)) = \begin{cases} +1 & \text{if } u_i(t) \geq 0 \\ -1 & \text{if } u_i(t) < 0 \end{cases} \quad (3.2)$$

where $u_i(t) = \sum_{j=1}^n w_{ij}v_j(t) - \theta_i$. Every node computes this operation in parallel until the network converges.

Hopfield have demonstrated that the Hopfield net can be used as a content addressable memory. Suppose we want to store m exemplars V^1, V^2, \dots, V^m . Let

$\mathcal{V} = [V^1|V^2|\dots|V^m]$, an $n \times m$ matrix. The outer-product algorithm defines the connection matrix as

$$W = \mathcal{V}\mathcal{V}^T - m \cdot I \quad (3.3)$$

where I is an $n \times n$ identity matrix. Thus, the matrix W is symmetric with zero diagonal elements. The basic idea is to construct the correlations between nodes. The larger absolute magnitude of w_{ij} , the stronger the correlation is between node i and node j . In this case, $\vec{\theta}$ is set equal to zero. When presented with a partial or noisy version of one of the stored exemplars, the network converges to that exemplar. However, the exemplars need to be nearly orthogonal to each other in order to be stable states of the network.

In the application of pattern recognition, neural networks based on the Hopfield net have been shown being able to perform classification of vowels and consonants extracted from spoken words [Gol86], lines in images [GS86], and digital character fonts [LGM87]. In addition, the Hopfield net has been successfully applied to optimization problems, such as the traveling salesperson problem [HT85], the A/D converter problem [TH86], the linear programming problem [TH86], and the matrix inverse problem.

The Hopfield net serves as a content addressable memory which retrieves one of the stored exemplars by giving a partial or noisy version of a stored exemplar as input. Essentially, a content addressable memory is a pattern classifier which classifies a pattern with the exemplar of the selected class instead of an index to the class. Lippmann et al. [LGM87] identified two necessary conditions for the Hopfield net to be used as a pattern classifier:

1. The exemplars of classes should be the stable states.
2. After convergence, a mechanism should be provided to determine which of

stored exemplars the current stable state of the net is closest to.

3.4.5 The Hamming Net

Lippmann et al. [LGM87] presented a Hamming net, which is a two-layer neural network and performs minimum Hamming distance classification. The Hamming distance is the number of different bits between two binary patterns in the corresponding bit positions. The first layer is made up of a feedforward perceptron network to calculate the likelihoods of the input to the stored exemplars. The second layer, called *maxnet*, is a laterally interconnected network that performs the winner-take-all function and determines which of its inputs is the maximum.

As its name implies, the Hamming net operates on binary patterns and calculates the likelihoods based on the Hamming distance. Instead of calculating the Hamming distance directly, the likelihoods of an input pattern to the stored exemplars is measured by N minus the Hamming distance, where N is the number of bits of the pattern vector. N minus the Hamming distance to the j -th exemplar can be obtained from a weighted sum of the elements of the input vector. That is,

$$N - N_{hamming}^j = c_j + \sum_{i=1}^N w_{ij}x_i, \quad (3.4)$$

where $w_{ij} = \frac{x_i^j}{2}$, $c_j = \frac{N}{2}$. In this equation x is a bipolar input vector and x^j denotes the j -th exemplar. Thus the first layer of the Hamming net is constructed by the connection weight w_{ij} 's and threshold $-c_j$'s.

The maxnet is a fully connected network consisting of k threshold logic nodes, where k is the number of classes. Each node feeds its output back to its input for excitatory, while it inhibits all other nodes via the lateral interconnections. Mathematically, the maxnet iteratively operates the following function to find the

maximum

$$v_i(t+1) = f\left(\sum_{j=1}^k T_{ij}v_j(t)\right), \quad (3.5)$$

where the connection weight

$$T_{ij} = \begin{cases} 1 & i = j \\ -w & (w < \frac{1}{k-1}) \text{ otherwise} \end{cases} \quad (3.6)$$

The threshold logic function $f(\cdot)$ is defined as follows:

$$f(u) = \begin{cases} u & \text{if } u \geq 0 \\ 0 & \text{if } u < 0 \end{cases} \quad (3.7)$$

The initial values $v_i(0)$, $i = 1, 2, \dots, k$, are the likelihoods obtained directly from the outputs of the first layer. That is the outputs of the first layer perceptron network are fed to the maxnet at time zero and then removed. Figure 3.2 shows the topology of a Hamming net.

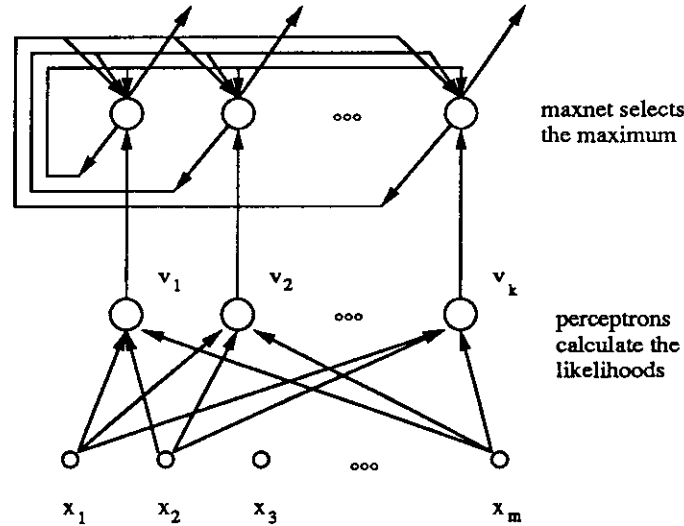


Figure 3.2: A Hamming net.

After convergence, the outputs stop changing and only the output of the node corresponding to the maximum input is positive. Therefore, the input pattern is

classified with the class indicated by the output node with a positive value. The proof of convergence of the maxnet was shown in [LGM87]. The basic idea of the proof is that the inhibition to the node with the maximum value is always less than the inhibition to all other nodes. It is also shown that the Hamming net outperforms the Hopfield net for pattern recognition applications in both ways of misclassification rate and number of interconnections.

3.5 Neural Networks for Pattern Recognition

A taxonomy of neural networks for pattern recognition is divided into *learning* and *programming* training schemes. Since the knowledge of a neural network is stored in its connection weights, the training schemes deal with the selections of the connection weights to accomplish the objective of the neural network. We define the learning process to be that the connection weights are adjusted gradually by presenting the training examples repeatedly. By programming, we mean that the connection weights can be directly programmed from the training examples.

There are also two subdivisions of the training schemes: *supervised* and *unsupervised* learning. Networks trained with supervision are provided with information that specifies the correct class for new input patterns during training. The Hopfield net and the Hamming net which can be programmed by the training set belong to the category of networks with the programming and supervised training scheme.

The multilayer perceptron is also trained with supervision, but its connection weights are adjusted via a learning process. Basically, it is trained to emulate the membership function in classification.

Networks trained without supervision, such as Kohonen's feature map and Grossberg's ART, do not need the information concerning the correct class during training. This type of networks can be used as vector quantizer or cluster

Table 3.1: A taxonomy of neural network for pattern recognition.

	supervised	unsupervised
learning	multilayer perceptron LVQ	Grossberg's ART Kohonen's feature maps
programming	Hopfield net Hamming net	–

formation.

Networks trained via a learning process present the capabilities of self organization and adaptation, while the networks with programmed connection weights do not have adaptability. On the other hand, the learning process is computationally intensive and not well understood. This type of networks becomes unreliable when the inputs are beyond the range of their training examples. Table 3.1 shows the taxonomy of neural network based on training algorithms.

In this study, we develop neural networks based on the programming and learning training schemes to accomplish the pattern recognition task. A neural network classifier is initially constructed with an architecture similar to the Hamming net and then tuned by a learning algorithm to improve its performance.

Part II

Methodology

CHAPTER 4

Pattern Designation

4.1 What is a Pattern?

The first step in any PR system is to define classes of patterns to be recognized. In many PR applications, patterns are formed by the nature of problems. For instance, a pattern may be a handprinted character or signals of a spoken word in image or speech recognition problems, respectively. Nevertheless, some patterns may be designated artificially in other applications. For example, the traffic at a street intersection can be defined as a pattern. The control of traffic lights may depend on the detection of certain traffic patterns in order to manage the traffic throughput. A pattern class is a set of patterns with certain identical or similar properties. These properties intend to characterize the objectives of applications.

Semantically, a “pattern” is the conceptual representation of an object in a PR application. On the other hand, a “pattern vector” is the physical representation of an objective for machine processing. A pattern of a handprinted character “8” may be labeled as “8”; while its pattern vector may be a bitmap or a binary vector to represent this specific handprinted character in a form suitable for machine processing. The distinction between the two terminologies is not so critical. As a matter of fact, they are interchangeable in this dissertation unless explicitly addressed elsewhere.

In the context of system identification problem domain, a pattern is considered as a candidate mathematical model modeling a real world system or a subsystem in the case of complex systems. The behavior of a pattern is represented

by the system response of its corresponding model. Its pattern vector is actually the system response unless some preprocessing is performed on the raw data for transformation or noise reduction. Therefore a pattern and a mathematical model are conceptually equivalent in this context. The space constructed by the pattern vector is referred to be the “pattern vector space” or “system response space.” However, a “parameter space” defined by the parametric model set is the conceptual representation of patterns. Given an input function and initial conditions, the model structure maps a model in the parameter space into a system response in the system response space.

4.2 What is Pattern Designation?

Pattern designation is intended to define a mathematical model as a pattern and to cluster patterns into classes. The criterion of grouping patterns into classes could be natural when patterns with similar system responses are grouped together, or artificial when patterns with the same mathematical form are grouped together. The criterion also strongly depends on the objective of characterizing a real world system and the available feature extraction techniques. If the feature extractor cannot extract the invariant properties of the patterns in the same class, there is no way to classify a given pattern. Similarly, it makes no sense if the pattern classes cannot achieve the objective of characterizing the system.

To express the concept formally, let us define the *parameter space* \mathcal{P} to be the space of the representation of various mathematical models that possibly model the system. Thus, for any pattern p , modeling the system, we have

$$p \in \mathcal{P} \tag{4.1}$$

A pattern class is a set of patterns with a certain property, that is,

$$P_i = \{p \mid p \in \mathcal{P}, \text{ and } p \text{ possesses property } i\}. \quad (4.2)$$

The *property* of class i can be defined to be the same mathematical form or similar system responses depending on the objectives of applications. Once the criterion of grouping classes is determined, the pattern classes are designated to be P_1, P_2, \dots , and P_k such that

$$P_i \subseteq \mathcal{P} \quad (i = 1, 2, \dots, k) \quad (4.3)$$

The designation is under the same input function and initial conditions for all patterns in \mathcal{P} . In addition, an extra class is defined to be

$$P_{k+1} = \mathcal{P} \cap \overline{(\cup_{i=1}^k P_i)} \quad (4.4)$$

The class P_{k+1} is a residue class used when observations of the system cannot be classified into any of the k predefined classes. Suppose that a pattern p_a is the best model in \mathcal{P} for modeling the real world system and $p_a \in P_i$. The observation should be classified as class i for a correct classification. Consequently, the parameter space \mathcal{P} can be reduced to P_i for the optimization approach in order to obtain a better initial structure instead of \mathcal{P} or for the PR approach to continue on a smaller parameter space.

Moreover, the classes can be designated to be Q_1, Q_2, \dots , and Q_l if the system can be observed under a new input function or different initial conditions. This different set of classes could provide more information in reducing the size of parameter space. If the second observation is classified as class Q_j , the new parameter space can then be further reduced to

$$\mathcal{P}_{new} = P_i \cap Q_j \quad (4.5)$$

For effective parameter space reduction and efficient classification, the classes should be made as disjoint as possible. Therefore, constructing the parameter space \mathcal{P} and partitioning \mathcal{P} into classes are the main issues in pattern designation. In section 4.3 and 4.4, domain discretization and partitioning parameter space scheme are proposed to achieve these tasks.

4.3 Domain Discretization

This research deals with mechanistic models rather than black box models because the former ones provide more insight into the system components. Based on deduction rules and a priori knowledge, a mechanistic model structure is derived for the system being modeled. Parameters of the mechanistic model structure may be functions of the system inputs and states. Some of the parameters may be unknown and must be determined from the observed data of the system. The objective of this research is to characterize these unknown parameters in order to build a complete mathematical model.

Suppose that the function $f(\theta)$ is an unknown parameter associated with this model structure, where θ denotes system inputs or state variables. It is necessary to establish a parametric form of this unknown function in order to identify its best approximation. A polynomial form of θ or some other mathematical form with undefined coefficients could represent this unknown function. Once these undefined coefficients are obtained, $f(\theta)$ is completely determined to some extent of approximation. So these coefficients construct a parameter space representing $f(\theta)$. Any particular function to represent $f(\theta)$ is designated to be a pattern. In fact, there are too many ways to specify the unknown function approximately. Also, different representations may coincide with the same function surface or curve. Different functions in the $(\theta, f(\theta))$ space may intersect or overlap each

other, where the $(\theta, f(\theta))$ space is defined to be a k dimensional space constructed by θ and $f(\theta)$ with $k = m + n$ (m and n denoting the dimensionality of θ and $f(\theta)$, respectively). In this case, it is difficult to partition the $(\theta, f(\theta))$ space into disjoint regions for designating pattern classes. Therefore, a unified, effective, and efficient representation of the unknown function is needed. The parameter space should provide a representation of pattern classes such that classes can be partitioned disjointly.

Besides, what is of interest in this study is to portray a general shape of the surface or curve of the unknown function $f(\theta)$ when a set of system responses is presented. The general shape of the surface or curve of the unknown function, also termed functional region, characterizes this function and also bounds the range of it for further investigation. So the possible functions to represent $f(\theta)$ that have similar shape in the $(\theta, f(\theta))$ space and also cause the models to generate similar responses should be grouped into the same class so as to utilize pattern recognition techniques for parameter characterization. Nevertheless, the argument is based on an assumption imposed in the study.

Assumption 4.1 *The models with similar parameters are likely to generate similar responses under the same input function and initial conditions. In other words, the system responses vary smoothly with smoothly varying parameters and do not behave chaotically.*

This assumption also implies that the system responses are bounded. This is always true for physical systems because of energy conservation. It is not necessary to consider the unstable models. The functions with similar shapes can be considered as neighbors. Of special importance here is to define a parameter space such that neighboring functions in the $(\theta, f(\theta))$ space are also neighbors in their representation space with a one to one mapping relation. In this case, the coher-

ence property exists within any pattern class in both the parameter space and the system response space. The model classes in the parameter space and the decision regions in the response space are thus tightly associated.

Two schemes are proposed to approximate the unknown function and define a parameter space:

1. *piecewise linear functions* and
2. *piecewise smooth polynomials*.

First of all, the continuous domain of θ is discretized into discrete elements. The value of $f(\theta)$ at each discrete point in the (θ) space becomes a variable in the function approximation, where the (θ) space denotes the m dimensional space constructed by θ , and m is the dimensionality of θ . In the piecewise linear function scheme, the function within each discrete element is approximated by a straight line or a hyperplane. In the other scheme, the function is approximated by a polynomial expression passing through the assigned values of $f(\theta)$ at all discrete points.

The number of discrete points in the (θ) space is the dimensionality of the parameter space. There is a tradeoff between the accuracy of the unknown function approximation and the dimensionality of the new parameter space. In general, the more discrete points, the better the approximation it is. The surface or curve of a function $f(\theta)$ in the $(\theta, f(\theta))$ space is then mapped to a point in the parameter space. Different functions in the $(\theta, f(\theta))$ space do not overlap or intersect in the parameter space. Disjoint regions can be effectively defined.

Example 4.1: Suppose that $f(x)$ is an unknown function and the interval of interest in the domain of $f(x)$ is $[x_0, x_3]$. Figure 4.1(a) shows a possible curve of $f(x)$. Figure 4.1(b) shows an example of the approximation of $f(x)$ by piecewise

linear functions. The parameter space is a four dimensional space constructed by $f(x_0)$, $f(x_1)$, $f(x_2)$, and $f(x_3)$. The positions of the discrete points, x_0 , x_1 , x_2 , and x_3 , can be determined according to the variability of $f(x)$. That is, more discrete points are needed to approximate $f(x)$ in the domain with larger variation of $f(x)$. If there exists no prior knowledge about the variability of $f(x)$, the domain can be divided into equal space. An approximated $f(x)$ is mapped to a point with coordinates $f(x_0), f(x_1), f(x_2), f(x_3)$ in the parameter space.

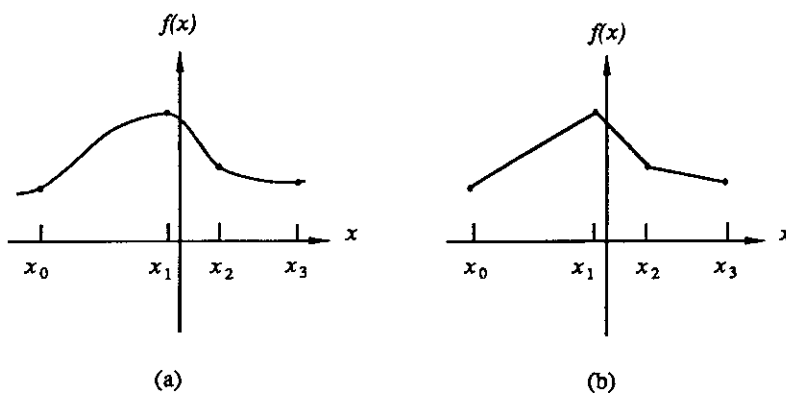


Figure 4.1: (a) A possible curve of the unknown function $f(x)$. (b) A piecewise linear approximate.

In the piecewise smooth polynomial scheme, $f(x)$ can be approximated by the polynomials

$$\hat{f}(x) = a + bx + cx^2 + dx^3. \tag{4.6}$$

This polynomial satisfies $\hat{f}(x_i) - f(x_i) = 0$, for $i = 0, 1, 2, 3$. Notice that the parameter space is the same as the one in the piecewise linear function scheme. The reason for selecting $f(x_0)$, $f(x_1)$, $f(x_2)$, and $f(x_3)$ as the parameter space instead of a , b , c , and d is to ensure that functions in the neighborhood in the $(x, f(x))$ space are also in the neighborhood in the parameter space defined by the vectors $(f(x_0), f(x_1), f(x_2), f(x_3))$. \square

Once the domain is discretized, the order of the polynomials or the number of

line segments (or hyperplane pieces) is determined in both schemes. This is similar to determining the mathematical structure in conventional system identification approaches. However, the objective of this study is to discriminate the general shape of the unknown functions rather than to identify the numerical values of unknown coefficients under the predetermined mathematical structure. In other words, the observation is classified with a region in the parameter space, not a point in the parameter space. Besides, this mathematical structure in either polynomials or piecewise linear functions is only an approximation of the unknown function. After a general shape for the unknown function is obtained, a more accurate approximation function can be constructed for the optimization or PR approach.

4.4 Partitioning Parameter Space

The partitioning of the parameter space deals with the separation of patterns into classes. This task greatly affects feature extraction and classification methods because of the rule used to separate patterns. A pattern class contains patterns whose corresponding models generate similar system responses to ensure the coherence property. This is the main purpose of partitioning parameter space as well as a direction in designing a feature extractor and a pattern classifier. The algorithm proposed to attack this problem is termed *top down dichotomy*. This algorithm divides the parameter space into disjoint contiguous regions according to the variation of various simulated system responses that are generated by various patterns sampled from the parameter space. Hence patterns of the same class have system responses varying within a certain threshold. In other words, there exists a certain degree of similarity among the patterns in one class. This property presents the coherence of a pattern class, and is useful for constructing the feature extractor

and classifier.

4.4.1 Algorithm 4.1: Top down dichotomy

Input: A mechanistic model structure with some unknown functions and a parameter space representing these unknown functions.

Output: A set of disjoint regions of the parameter space as well as a between-class covariance matrix and a within-class covariance matrix of the simulated system responses of the sample patterns.

1. Start with the whole parameter space as one region and an initial number of sample patterns in this region $s_1 = 0$, where s_1 denotes the current number of sample patterns in this region.
2. For a region in the parameter space, randomly select $s - s_1$ patterns with uniform distribution, where s is a predetermined number of total sample patterns in a region.
3. Run computer simulations of these $s - s_1$ patterns, and combine with the existing simulation results of s_1 patterns to get s sets of simulated system responses.
4. Calculate the *variation* of all these s response vectors whose variance could be used as a metric for measuring the variation.
5. Repeat step 2 to step 4 for all regions in the parameter space.
6. If there exists a region with variation exceeding a certain threshold, this region is split into two. Otherwise, calculate the covariance matrices and stop.

7. Calculate s_1 , the current number of sample patterns, in these two new regions. Go to step 2 and handle each region independently and recursively.

In this algorithm, the parameter s is chosen such that all s patterns in a region are able to characterize the variation property of this region. Assumption 4.1 guarantees the algorithm to stop. If the system response is not bounded, the algorithm will not stop and will keep partitioning the parameter space where the system response is unstable.

4.4.2 Discussion and Analysis of Algorithm 4.1

In step 2, the first problem encountered is “how many sample patterns in a region would be enough to portray the variation of this region?” There exists no theoretical proof on the number of sample patterns required. Under this situation, the number of sample patterns becomes a tradeoff between the accuracy of the variation and the amount of computation required for simulating all sample patterns. The other problem is the method of sampling. Although the patterns are sampled randomly from the parameter space, these patterns are not distributed uniformly when the number of sample patterns, s , is not large enough. Their capability to characterize a region becomes weak if they are not evenly distributed in this region. Thus, one of the sampling techniques employed in this study is *stratified random* sampling [Coc77], which means the region is equally divided into s subregions, and one pattern is randomly sampled from each subregion. In addition, *systematic* sampling technique [Coc77] is also used because it is deterministic and easy to analyze. These two sampling schemes spread samples more evenly over a sampling space. More analysis details are discussed in section 4.5.

In step 4, the variation can be defined in terms of the variance of sample patterns’ system responses in this region or the longest distance from any system

response to the mean response vector. The variance can be defined as the scatter or covariance matrix of the system responses. For simplicity, the trace of the covariance matrix is used as the scalar measure of variation. Both metric schemes, the variance or the longest distance, are based on the concept of Euclidean distance. Hence, the results of partitioning via both schemes are similar if not identical.

In step 6, the major concern is the dimension in the parameter space where the partition takes place. The dimensions in the parameter space are the variables representing the unknown parameter and constructing the parameter space. This dimension should be the one for which the corresponding variable's variation causes the largest variation in system responses. Based on this idea, every dimension is assumed to be a candidate and is presumably split at the middle to obtain two subregions. The distance between the mean response vectors of the two subregions is calculated. Then the dimension with the longest distance is actually split. A theoretical proof on the selection of the splitting dimension is given in Appendix A.

The threshold used in step 6 is adjustable. Basically, it is adjusted from a larger to a smaller value if the number of regions obtained after the algorithm terminates is not large enough.

Because of the partitioning property, the sensitivity of the system responses to the parameters is embedded in the partitioned regions. The parameter space, with more sensitive system responses to parameters, is eventually divided into a smaller region; while a larger region means that the system responses are less sensitive to the variation of the parameters.

In step 5, every region can be processed independently and in parallel. Even the procedures in one region, such as step 2 and 3, can be processed in parallel for all sample patterns. Data communications are required only to compute the variation of system responses and to determine the splitting dimension of the parameter

space. This algorithm is, therefore, well suited for parallel implementation.

4.5 Sample Size Determination

Sampling techniques are employed to study the characteristics of a continuous function $z(x)$ over a domain space Q , especially, when an analytical closed form for $z(x)$ does not exist. Another reason for using sampling techniques is to utilize digital computers to evaluate $z(x)$. Because the elements of the continuous domain are uncountable and a complete count is impossible, discretization and sampling is always necessary. Estimating the mean or variance of $z(x)$ is often the purpose of sampling. In this study, variance estimation is the major concern as mentioned in Algorithm 4.1, where the parameter space is the domain space Q and the system response of a model structure with the parameter x is the function $z(x)$.

There are two important issues concerning sampling techniques. One is the sampling error. The other is the cost of sampling, which includes computation and data collection etc. The tradeoff is between cost and accuracy. The larger the sample size, the greater is the cost of evaluating the sample points and collecting data. The objective here is to study sample size determination in order to attain a specified precision while minimizing the cost.

Three sampling schemes are discussed below. An approach to analyzing the sampling performance is given so as to determine the best sample size under the same sampling scheme.

4.5.1 Sampling Schemes

The simplest type of sampling is *simple random sampling*. In this case, each sample point is selected with a uniform distribution over the whole sample space, and the selection is independent of the other sample points. When the sample size is

small, this sampling process usually does not cover the entire space uniformly. The sampling performance becomes poor. Thus, this sampling scheme is not employed in this study.

The second type of sampling is *stratified random sampling*. The space Q is partitioned into disjoint subspaces of Q_1, Q_2, \dots, Q_l with sample sizes N_1, N_2, \dots, N_l , respectively. These subspaces are called *strata*. A simple random sampling is then taken in each stratum. This scheme ensures that the sample points are spread more evenly over the space Q .

The third type of sampling is *systematic sampling*. A lattice network is superimposed on the space Q . Points sampled inside each lattice form a regular geometric configuration and all the lattices have the same sampling configuration. This scheme can be viewed as a stratified sampling with an equal size of strata except those on the border; and a regular sampling configuration is conducted in each stratum. If $z(x)$ is a periodic function in its domain space, the regular sampling configuration of systematic sampling may coincide with the period of $z(x)$ and fails to capture the properties of the function.

4.5.2 The Efficiency of Sampling

Since we are interested in estimating the mean or variance of $z(x)$, let us define the true mean of $z(x)$ in the space Q as

$$\bar{z} = \frac{\int_Q z(x) dx}{\int_Q dx}, \quad (4.7)$$

the estimated mean of n sample points as

$$\hat{z}_{(n)} = \frac{1}{n} \sum_{i=1}^n z(x_i), \quad (4.8)$$

the true variance as

$$\sigma^2 = \frac{\int_Q (z(x) - \bar{z})^2 dx}{\int_Q dx}, \quad (4.9)$$

and the estimated variance of n sample points as

$$\hat{\sigma}^2_{\langle n \rangle} = \frac{1}{n} \sum_{i=1}^n (z(x_i) - \hat{z}_{\langle n \rangle})^2, \quad (4.10)$$

where the integral \int_Q denotes the integration of x over the space Q and $z(x_i)$ is the value of z sampled at x_i . Also, denote

$$\|Q\| \triangleq \int_Q dx \quad (4.11)$$

as the *size* of Q . For instance, $\|Q\|$ is the length, area, or volume of Q if Q is a one, two, or three dimensional space, respectively.

It is obvious that a good sampling scheme should use a small sample size while leading to a low sampling error of mean estimation, $\hat{z}_{\langle n \rangle} - \bar{z}$. In general, the variance

$$E[(\hat{z}_{\langle n \rangle} - \bar{z})^2] \quad (4.12)$$

is used as a measurement to characterize the sampling performance [Mat80]. Once the required precision of estimations is specified, the sampling performance can be used to determine the sample size. Let us assume that

1. A larger sample size results in a more accurate estimation as long as the sample points are evenly distributed over the function domain. The assumption is good only if the function is bounded inside the space Q . The idea of increasing the sample size until achieving the desired accuracy is based on this assumption.
2. The “border effect” is negligible. In the case of a systematic sampling scheme, every lattice unit or stratum is assumed to be of equal size. This means that the lattice units along the border are of the same size as those lattice inside the space, and have the same sampling effects as well.

4.5.3 Estimating the Sampling Error of Systematic Sampling in One Dimension Space

Consider a sampling scheme in which only one point is sampled from each lattice. That is, the domain space is divided into n equal subspaces, Q_1, Q_2, \dots, Q_n , where $\|Q_1\| = \|Q_2\| = \dots = \|Q_n\| = \|\Delta Q\|$. And the sample point x_i is at the center of Q_i . Hence, n is the number of sample points and $\|\Delta Q\|$ is the size of lattice unit in the systematic sampling scheme.

The analysis of sampling errors is based on the Taylor expansion of $z(x)$. Sampling errors can be obtained analytically in terms of the second derivative of $z(x)$ and the size of the lattice unit. However, the analytical closed form of $z(x)$ is generally not available. The estimations of sampling errors are then carried out using two sampling results.

4.5.3.1 Sampling Error of Mean Estimations

Because of the deterministic sampling scheme, the sampling error, $\hat{z}_{(n)} - \bar{z}$, can be used to characterize the sampling performance of mean estimations.

$$\begin{aligned}
 \hat{z}_{(n)} &= \frac{1}{n} \sum_{i=1}^n z(x_i) \\
 &= \frac{1}{n\|\Delta Q\|} \sum_{i=1}^n \|\Delta Q\| z(x_i) \\
 &= \frac{1}{\|Q\|} \sum_{i=1}^n \|\Delta Q\| z(x_i), \tag{4.13}
 \end{aligned}$$

Thus, the sampling error yields

$$\frac{1}{\|Q\|} \left(\sum_{i=1}^n \|\Delta Q\| z(x_i) - \int_Q z(x) dx \right) = \frac{1}{\|Q\|} \sum_{i=1}^n \varepsilon_i, \tag{4.14}$$

where $\varepsilon_i = \|\Delta Q\| z(x_i) - \int_{Q_i} z(x) dx$.

Let $Q_i = [a_i, b_i]$ in the one dimension case and then $x_i = (a_i + b_i)/2$. By Taylor

expansion at $x = a_i$, the function has the form

$$z(x) = z(a_i) + (x - a_i)z'(a_i) + \frac{(x - a_i)^2}{2!}z''(a_i) + \frac{(x - a_i)^3}{3!}z'''(a_i) + \dots \quad (4.15)$$

Taking integration on the both sides of the above equation in the interval $[a_i, b_i]$, we have

$$\int_{a_i}^{b_i} z(x)dx = (b_i - a_i)z(a_i) + \frac{(b_i - a_i)^2}{2!}z'(a_i) + \frac{(b_i - a_i)^3}{3!}z''(a_i) + \dots \quad (4.16)$$

Also,

$$\begin{aligned} z(x_i) &= z\left(a_i + \frac{(b_i - a_i)}{2}\right) \\ &= z(a_i) + \left(\frac{b_i - a_i}{2}\right)z'(a_i) + \left(\frac{b_i - a_i}{2}\right)^2 z''(a_i)/2! + \dots \end{aligned} \quad (4.17)$$

Then

$$\|\Delta Q_i\|z(x_i) = (b_i - a_i)z(a_i) + \frac{(b_i - a_i)^2}{2}z'(a_i) + \frac{(b_i - a_i)^3}{8}z''(a_i) + \dots \quad (4.18)$$

Therefore,

$$\begin{aligned} \varepsilon_i &= -\frac{1}{24}(b_i - a_i)^3 z''(a_i) + r_i(b_i - a_i)^4 z'''(\xi_i) \\ &= -\frac{\|\Delta Q\|^3}{24}z''(a_i) + O(\|\Delta Q\|^4), \end{aligned} \quad (4.19)$$

where $r_i(b_i - a_i)^4 z'''(\xi_i)$ is a residual term with a constant r_i and $\xi_i \in [a_i, b_i]$.

Equation (4.14) becomes

$$\begin{aligned} &\frac{1}{\|Q\|} \sum_{i=1}^n \left(-\frac{\|\Delta Q\|^3}{24}z''(a_i) + O(\|\Delta Q\|^4)\right) \\ &\leq \frac{1}{\|Q\|} \sum_{i=1}^n \left(-\frac{\|\Delta Q\|^3}{24}z''_{\min}(x) + O(\|\Delta Q\|^4)\right) \\ &= -\frac{\|\Delta Q\|^2}{24}z''_{\min}(x) + O(\|\Delta Q\|^3), \end{aligned} \quad (4.20)$$

where $z''_{\min}(x)$ is the minimum of $z''(x)$ for all $x \in Q$. By a similar argument, the sampling error is bounded from below by

$$-\frac{\|\Delta Q\|^2}{24}z''_{\max}(x) + O(\|\Delta Q\|^3), \quad (4.21)$$

where $z''_{max}(x)$ is the maximum of $z''(x)$ for all $x \in Q$.

Finally, we can obtain the error of the mean estimation of n sample points

$$\epsilon\langle n \rangle = \frac{1}{\|Q\|} \sum_{i=1}^n \epsilon_i = -\frac{\|\Delta Q\|^2}{24} z''(\xi) + O(\|\Delta Q\|^3), \quad (4.22)$$

by mean value theorem provided that $z''(x)$ is continuous in Q and $\xi \in Q$.

However, the function $z(x)$ is usually unknown, as well as its second derivative, $z''(x)$. In order to estimate the sampling error, another systematic sampling with $2n$ sample points is conducted and the corresponding sampling error is

$$\epsilon\langle 2n \rangle = -\frac{\|\Delta Q\|^2}{96} z''(\zeta) + O\left(\left\|\frac{\Delta Q}{2}\right\|^3\right), \quad (4.23)$$

for some $\zeta \in Q$. Incorporating the sampling error term, we have

$$\hat{z}\langle n \rangle = \bar{z} + \epsilon\langle n \rangle \quad (4.24)$$

and

$$\hat{z}\langle 2n \rangle = \bar{z} + \epsilon\langle 2n \rangle, \quad (4.25)$$

where $\hat{z}\langle n \rangle$ and $\hat{z}\langle 2n \rangle$ are the mean estimations of n and $2n$ sample points, respectively. Assume that the second derivative of $z(x)$ does not change much in the sampling interval, that is, $z''(a_i) \simeq z''(a_{i2})$, where $a_{i2} = (a_i + b_i)/2$ for halving the sampling interval. Thus, we have $z''(\xi) \simeq z''(\zeta)$ and the estimation of sampling error with $2n$ sample points is

$$\widehat{\epsilon\langle 2n \rangle} \approx \frac{1}{3}(\hat{z}\langle n \rangle - \hat{z}\langle 2n \rangle) \quad (4.26)$$

4.5.3.2 Sampling Error of Variance Estimations

To estimate the sampling error of variance, let us examine $\hat{\sigma}^2\langle n \rangle - \sigma^2$. By Equations (4.9) and (4.10), we have

$$\sigma^2 = \frac{\int_Q z^2(x) dx}{\|Q\|} - \bar{z}^2 \quad (4.27)$$

and

$$\hat{\sigma}^2_{\langle n \rangle} = \frac{1}{n} \sum_{i=1}^n z^2(x_i) - \hat{z}_{\langle n \rangle}^2. \quad (4.28)$$

So

$$\hat{\sigma}^2_{\langle n \rangle} - \sigma^2 = \frac{1}{\|Q\|} \left[\sum_{i=1}^n \|\Delta Q\| z^2(x_i) - \int_Q z^2(x) dx \right] - (\hat{z}_{\langle n \rangle}^2 - \bar{z}^2). \quad (4.29)$$

Let $g(x) = z^2(x)$ and we can obtain

$$\frac{1}{\|Q\|} \left[\sum_{i=1}^n \|\Delta Q\| z^2(x_i) - \int_Q z^2(x) dx \right] = -\frac{\|\Delta Q\|^2}{24} g''(\eta) + O(\|\Delta Q\|^3) \quad (4.30)$$

for some $\eta \in Q$. Also from Equation (4.24),

$$\begin{aligned} \hat{z}_{\langle n \rangle}^2 - \bar{z}^2 &= (\bar{z} + \epsilon(n))^2 - \bar{z}^2 \\ &= 2\epsilon(n)\bar{z} + (\epsilon(n))^2. \end{aligned} \quad (4.31)$$

Hence, the sampling error of the variance estimation of n sample points is approximately four times as large as that of $2n$ sample points. Thus, we can estimate the sampling error of variance estimation as

$$\epsilon_{\sigma^2}(\widehat{2n}) \approx \frac{1}{3}(\hat{\sigma}^2_{\langle n \rangle} - \hat{\sigma}^2_{\langle 2n \rangle}). \quad (4.32)$$

Using Equations (4.26) and (4.32), we can determine the sample size to achieve the desired accuracy of estimations. This technique doubles the sample size and estimates the sampling error to see if it is smaller than a tolerable error.

A similar analysis can be applied to estimate the sampling error in a higher dimensional space Q . For example, suppose that $Q \subseteq \mathbb{R}^2$, and h and k are the discrete intervals of the lattice unit in the dimensions x and y , respectively. Then the sampling error of mean estimation is

$$-\frac{1}{24}(h^2 z''_{xx}(\xi, \varphi) + k^2 z''_{yy}(\xi, \varphi)) + O(h^3 + h^2 k + h k^2 + k^3), \quad (4.33)$$

for some $(\xi, \varphi) \in Q$. The sampling error of n sample points can be estimated as

$$\widehat{\epsilon(n)} \approx 2(\hat{z}_{(n)} - \hat{z}_{(2n)}) \quad (4.34)$$

from the doubled sample size of $2n$ by reducing h and k by a factor of $\sqrt{2}$ or

$$\widehat{\epsilon(n)} \approx \frac{4}{3}(\hat{z}_{(n)} - \hat{z}_{(4n)}) \quad (4.35)$$

from the quadrupled sample size of $4n$ which doubles the sample size in each dimension. Moreover, we can find that

$$\widehat{\epsilon(n)} \approx 2\widehat{\epsilon(2n)} \approx 4\widehat{\epsilon(4n)}. \quad (4.36)$$

when Q is a two dimensional space.

4.5.4 Estimating the Sampling Error of Stratified Random Sampling with One Sample Point Per Stratum

Assume that every stratum is of equal size in the following discussion.

4.5.4.1 Sampling Error of Mean Estimations

First, we show that the expected value of mean estimation is equal to the true mean.

Theorem 4.1 *The mean estimation via a stratified random sampling with one sample point per stratum is unbiased.*

Proof: Consider whether the mean sampling error $E[\hat{z}_{(n)} - \bar{z}] = 0$ or not. Let us examine the expectation of $\hat{z}_{(n)}$ in Equation (4.13).

$$\begin{aligned} E[\hat{z}_{(n)}] &= \frac{1}{\|Q\|} \sum_{i=1}^n \|\Delta Q\| E[z(x_i)] \\ &= \frac{1}{\|Q\|} \sum_{i=1}^n \|\Delta Q\| \int_{a_i}^{b_i} \frac{1}{\|Q_i\|} z(x_i) dx_i \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{\|Q\|} \sum_{i=1}^n \int_{a_i}^{b_i} z(x) dx \\
&= \bar{z}
\end{aligned} \tag{4.37}$$

because x_i has a uniform distribution in the interval $[a_i, b_i]$.

It follows that the mean estimation is unbiased. \square

Hence we need to employ the variance $E[(\hat{z}_{\langle n \rangle} - \bar{z})^2]$ to evaluate the sampling performance. By the above theorem, we have

$$\begin{aligned}
E[(\hat{z}_{\langle n \rangle} - \bar{z})^2] &= E[\hat{z}_{\langle n \rangle}^2] - 2E[\hat{z}_{\langle n \rangle}]\bar{z} + \bar{z}^2 \\
&= E[\hat{z}_{\langle n \rangle}^2] - \bar{z}^2
\end{aligned} \tag{4.38}$$

Recall that $\hat{z}_{\langle n \rangle} = \frac{1}{n} \sum_{i=1}^n z(x_i)$. $E[\hat{z}_{\langle n \rangle}^2]$ becomes

$$\begin{aligned}
&\frac{1}{n^2} E\left[\left(\sum_{i=1}^n z(x_i)\right) \cdot \left(\sum_{i=1}^n z(x_i)\right)\right] \\
&= \frac{1}{n^2} \left(\sum_{i=1}^n \frac{\int_{Q_i} z^2(x) dx_i}{\|Q_i\|} + \sum_i \sum_{j \neq i} \frac{\int_{Q_i} z(x_i) dx_i}{\|Q_i\|} \cdot \frac{\int_{Q_j} z(x_j) dx_j}{\|Q_j\|} \right)
\end{aligned} \tag{4.39}$$

because x_i and x_j are independent with respect to Q_i and Q_j . Since \bar{z} can be calculated from n disjoint subspaces, we get

$$\begin{aligned}
\bar{z}^2 &= \left(\frac{\sum_{i=1}^n \int_{Q_i} z(x) dx}{\|Q\|} \right)^2 \\
&= \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n \frac{\int_{Q_i} z(x) dx}{\|\Delta Q\|} \frac{\int_{Q_j} z(x) dx}{\|\Delta Q\|}.
\end{aligned} \tag{4.40}$$

Subtracting Equation (4.40) from Equation (4.39), $E[(\hat{z}_{\langle n \rangle} - \bar{z})^2]$ yields

$$\begin{aligned}
&\frac{1}{n^2} \sum_{i=1}^n \frac{\int_{Q_i} z^2(x) dx}{\|Q_i\|} - \frac{1}{n^2} \sum_{i=1}^n \frac{\int_{Q_i} z(x) dx}{\|\Delta Q\|} \cdot \frac{\int_{Q_i} z(x) dx}{\|\Delta Q\|} \\
&= \frac{1}{n^2} \sum_{i=1}^n \bar{z}_i^2 - \frac{1}{n^2} \sum_{i=1}^n \bar{z}_i^2 \\
&= \frac{1}{n^2} \sum_{i=1}^n \sigma_i^2,
\end{aligned} \tag{4.41}$$

where the subscript i indexes the statistics of the subspace Q_i . It is easy to see that

$$E[(\hat{z}_{(n)} - \bar{z})^2] = \begin{cases} \sigma^2 & \text{if } n = 1 \\ 0 & \text{if } n \rightarrow \infty. \end{cases} \quad (4.42)$$

If the subspace Q_i is equally divided into two subspace Q_{i1} and Q_{i2} and the sample size is doubled from n to $2n$, we can obtain the following relation

$$\sigma_i^2 = \frac{1}{2}(\sigma_{i1}^2 + \sigma_{i2}^2) + \frac{1}{4}(z_{i1} - z_{i2})^2. \quad (4.43)$$

Thus, the sampling performance given $2n$ sample points is

$$\begin{aligned} & E[(\hat{z}_{(2n)} - \bar{z})^2] \\ &= \frac{1}{4n^2} \sum_{i=1}^n (\sigma_{i1}^2 + \sigma_{i2}^2) \\ &= \frac{1}{2n^2} \sum_{i=1}^n [\sigma_i^2 - \frac{1}{4}(z_{i1} - z_{i2})^2] \\ &= \frac{1}{2} E[(\hat{z}_{(n)} - \bar{z})^2] - \frac{1}{8n^2} \sum_{i=1}^n (z_{i1} - z_{i2})^2 \\ &\leq \frac{1}{2} E[(\hat{z}_{(n)} - \bar{z})^2]. \end{aligned} \quad (4.44)$$

In other words, doubling the sample size improves the sampling performance more than twice.

Applying the same technique, it is easy to show that the variance of mean estimations is equal to one n^{th} of the true variance of the function z if the *simple random sampling* scheme is taken over Q . That is,

$$E[(\hat{z}_{(n)} - \bar{z})^2] = \frac{\sigma^2}{n}, \quad (4.45)$$

where n is the number of sample points. As n gets larger, the distribution of $\hat{z}_{(n)}$ approaches a normal distribution with mean \bar{z} and variance $\frac{\sigma^2}{n}$.

Suppose that the space Q is equally divided into n subspaces and stratified random sampling is conducted. We can find that

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n \sigma_i^2 + \frac{1}{n} \sum_{i=1}^n \bar{z}_i^2 - \bar{z}^2. \quad (4.46)$$

Plugging in Equation (4.41), we get

$$\sigma_{\hat{z}_{(n)}}^2 = E[(\hat{z}_{(n)} - \bar{z})^2] = \frac{\sigma^2}{n} - \frac{1}{n} \left(\frac{1}{n} \sum_{i=1}^n \bar{z}_i^2 - \bar{z}^2 \right). \quad (4.47)$$

By Cauchy inequality,

$$\frac{1}{n} \sum_{i=1}^n \bar{z}_i^2 \geq \left(\frac{1}{n} \sum_{i=1}^n \bar{z}_i \right)^2 = \bar{z}^2. \quad (4.48)$$

Therefore, the second term on the right hand side of Equation (4.47) is positive. Equations (4.45) and (4.47) illustrate that stratified random sampling obtains better sampling performance than simple random sampling, given the same number of sample points. By Chebyshev's inequality, the probability

$$P \left\{ |\hat{z}_{(n)} - \bar{z}| \geq k \right\} \leq \frac{\sigma_{\hat{z}_{(n)}}^2}{k^2} = \frac{\sigma^2 - c}{nk^2}, \quad (4.49)$$

where $c = \frac{1}{n} \sum_{i=1}^n \bar{z}_i^2 - \bar{z}^2$. Given a bound probability α and an interval k , we can solve the number of sample points

$$n = \frac{\sigma^2 - c}{\alpha k^2}. \quad (4.50)$$

Using the biased estimation of variance (see Equation (4.55)), the number of sample points is

$$n \approx \frac{\hat{\sigma}_{(n)}^2}{\alpha k^2} \quad (4.51)$$

in order to obtain the desired sampling performance. Thus we can obtain the probability less than α that the true mean \bar{z} is outside the interval $[\hat{z}_{(n)} - k, \hat{z}_{(n)} + k]$ given n sample points for the stratified random sampling scheme. This result can be generalized to vector function z provided that the operator $|\cdot|$ calculates Euclidean distance, and variances are also based on the Euclidean distances among vectors. In the multidimensional space, $|\hat{z}_{(n)} - \bar{z}| = k$ defines a hypersphere with radius k .

4.5.4.2 Sampling Error of Variance Estimations

In the following, let us examine the mean sampling error of the variance estimation of n sample points

$$\begin{aligned}
& E[\hat{\sigma}^2_{\langle n \rangle} - \sigma^2] \\
&= E\left[\frac{1}{n} \sum_{i=1}^n z^2(x_i) - \hat{z}^2_{\langle n \rangle}\right] - \left(\frac{\int_Q z^2(x) dx}{\|Q\|} - \bar{z}^2\right) \\
&= \frac{1}{n} \sum_{i=1}^n E[z^2(x_i)] - \frac{\int_Q z^2(x) dx}{\|Q\|} - (E[\hat{z}^2_{\langle n \rangle}] - \bar{z}^2). \tag{4.52}
\end{aligned}$$

Because x_i is uniformly distributed in Q_i , we can derive

$$E[z^2(x_i)] = \frac{\int_{Q_i} z^2(x) dx_i}{\|Q_i\|} \tag{4.53}$$

and

$$\frac{1}{n} \sum_{i=1}^n E[z^2(x_i)] = \frac{\int_Q z^2(x) dx}{\|Q\|}. \tag{4.54}$$

Thus, the expectation of sampling errors of the variance estimations becomes

$$\begin{aligned}
& E[\hat{\sigma}^2_{\langle n \rangle} - \sigma^2] \\
&= -(E[\hat{z}^2_{\langle n \rangle}] - \bar{z}^2) \\
&= -\frac{1}{n^2} \sum_{i=1}^n \sigma_i^2. \tag{4.55}
\end{aligned}$$

It is obvious that $\hat{\sigma}^2_{\langle n \rangle}$ is a biased estimation of σ^2 . Again, we can obtain the two extreme sampling situations

$$E[\hat{\sigma}^2_{\langle n \rangle} - \sigma^2] = \begin{cases} -\sigma^2 & \text{if } n = 1 \\ 0 & \text{if } n \rightarrow \infty. \end{cases} \tag{4.56}$$

Similarly, the expected sampling error of the variance estimation reduces in a factor greater than two when the sample size is doubled.

From Equations (4.41) and (4.55), it follows that

$$E[(\epsilon_{\langle n \rangle})^2] = \frac{1}{n^2} \sum_{i=1}^n \sigma_i^2 \tag{4.57}$$

and

$$E[\epsilon_{\sigma^2}(n)] = -\frac{1}{n^2} \sum_{i=1}^n \sigma_i^2. \quad (4.58)$$

Since the analytical form of $z(x)$ is in general unknown, the sampling errors which depend on σ_i^2 can only be obtained by estimation. The double-sample size technique is again employed to estimate the sampling errors. Assume that $\bar{z}_{i1} \simeq \bar{z}_{i2}$ for some sample size n . In this case, we have

$$E[(\epsilon(n))^2] \approx 2E[(\epsilon(2n))^2] \quad (4.59)$$

and

$$E[\epsilon_{\sigma^2}(n)] \approx 2E[\epsilon_{\sigma^2}(2n)]. \quad (4.60)$$

By doubling sample size, the expectations of sampling errors are

$$E[(\epsilon(2n))^2] \approx E[\hat{z}_{(n)}^2 - \hat{z}_{(2n)}^2] \quad (4.61)$$

and

$$E[\epsilon_{\sigma^2}(2n)] \approx E[\hat{\sigma}_{(n)}^2 - \hat{\sigma}_{(2n)}^2]. \quad (4.62)$$

4.6 Summary

Pattern designation involves the formulation of parameter characterization as a pattern recognition problem and leads to the designation of distinct nonlinear mathematical models to be patterns. A parameter space is defined to represent the candidate models by piecewise linear functions or piecewise smooth polynomials, which approximate the unknown functions. This parameter space is partitioned into pattern classes using a top down dichotomy algorithm based on the variation of simulated system responses of the candidate models. An analysis of sample size determination for this algorithm is proposed. The primary idea is to double the sample size and estimate the sampling error from the current estimation and the

previous estimation. Thus an approximate sample size can be determined for a desired accuracy.

CHAPTER 5

Feature Extraction

5.1 Introduction

Features are the variables that are measured and utilized by a classifier to separate patterns into classes. The basic requirements of the features selected are [Che73]:

1. they are invariant to the patterns in the same class,
2. they properly abstract the properties of a pattern,
3. they can be obtained from patterns.

The task of determining these variables from patterns is called *feature extraction*. This is the most important step in designing an efficient PR system. In general, features must be able to enhance the similarity of patterns within one class and at the same time enhance the difference of patterns between classes. The minimization of the probability or cost of misclassification are also criteria in selecting features.

Because of the importance that features can provide significant differences from one class to another, this subject has received much attention. Vigorous efforts were exerted in this field in late 1960's and early 1970's. Many important theories were derived and experiences were gained. But, no significant break through has been reported in both statistical pattern recognition and syntactic pattern recognition. In a survey paper, Levine [Lev69] commented that no general theory exists

for feature extraction. It turns out that feature extraction is very much *ad hoc* and problem oriented. For example, in order to identify the exponential decay of an underdamped sinusoidal signal, the peak point of every signal cycle is the best feature. This is because the system is well understood. In other words, the more prior knowledge about the problem, the easier the task. This kind of features depends heavily on human expertise. However, engineers and scientists sometimes cannot express clearly the features of applications in a formal language or mathematical expressions. Some feature extraction schemes are beyond the computational power of current computer technology or cannot be formulated as suitable algorithms to utilize computers. All of this make this task even tougher.

5.2 Objective

Although there exists no unified formulation of the feature extraction problem, the experience and theories obtained two decades ago are none-the-less valuable to this work. We can define the objective of feature extraction as general guidelines:

- To enhance the clustering within any one class,
- To increase the separation between classes,
- To reduce dimensionality of data.

In the present research we employ principal component analysis techniques as the general idea to extract the primitive features which fulfill the above objectives. The quest for primitive features has been always a most demanding challenge in diverse areas of science inquiry. In the early years, chemists tried to find the primitive elements composing materials. In the context of pattern recognition, the primitive features of a pattern would also play an important role. In the following, feature extraction techniques based on principal component analysis are discussed

because principal component analysis provides a statistical approach to analyze the primitive features.

5.3 Karhunen-Loève Expansion

One of the best known and most useful feature extraction techniques is the discrete Karhunen-Loève (K-L) expansion. The K-L expansion is a kind of principal component analysis. Basically, it is a linear transformation based on the eigenvectors of the covariance matrix of samples.

Let X be an n -dimensional random vector, or a system response vector in the context of the system identification problem domain. There exists an orthonormal expansion of the random vector X in the form

$$X = \sum_{i=1}^n y_i \Phi_i = \Phi Y \quad (5.1)$$

where Φ is an $n \times n$ matrix

$$\Phi = (\Phi_1 \cdots \Phi_n), \quad (5.2)$$

$$Y = (y_1 \cdots y_n)^T, \quad (5.3)$$

y_i is the coefficient of Φ_i in the expansion, and T is a transpose operator. The requirement of the orthonormal expansion is that the columns of Φ are orthonormal.

That is,

$$\Phi_i^T \Phi_j = \begin{cases} 1 & (i = j) \\ 0 & (i \neq j) \end{cases} \quad (5.4)$$

Consider the autocovariance matrix of X

$$\begin{aligned} \Sigma_X &= E[(X - M_X)(X - M_X)^T] \\ &= E[(\Phi Y - \Phi M_Y)(\Phi Y - \Phi M_Y)^T] \\ &= E[\Phi(Y - M_Y)(Y - M_Y)^T \Phi^T] \\ &= \Phi \Sigma_Y \Phi^T \end{aligned} \quad (5.5)$$

It has been shown that the autocovariance matrix Σ_Y must be a diagonal matrix Λ in order to minimize the mean square error of the expansion when fewer than n basis vectors are used [Fuk72]. That is,

$$\Sigma_X \Phi = \Lambda \Phi. \quad (5.6)$$

where

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \lambda_n \end{pmatrix} \quad (5.7)$$

The matrix Φ consists of n linearly independent column vectors, which are the eigenvectors of the autocovariance matrix of X . In other words, X can be expanded in the eigenvectors and Y is a linear transformation of X in the form

$$Y = \Phi^T X \quad (5.8)$$

The components of Y are considered to be the features of the observed vector X .

There are several attractive properties of the discrete K-L expansion.

1. The eigenvalues can be ordered in the following manner

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0. \quad (5.9)$$

The magnitude of an eigenvalue is the variance of X in the direction of its associated eigenvector and also represents the effectiveness of the feature component. Therefore, the reduction of the dimensionality of the features can begin by the deletion of the feature with the smallest eigenvalue. Usually, the criterion for selecting feature components is based on how much information is to be preserved after the orthonormal transformation. That is,

$$\min_m \frac{\sum_{i=1}^m \lambda_i}{\sum_{i=1}^n \lambda_i} > \rho. \quad (5.10)$$

where ρ is a preselected threshold to retain the amount of information during the transformation. Then m components of Y are selected with the m largest eigenvalues to be the features of X . In this case, the dimensionality is reduced from n to m .

2. The Euclidean distance is preserved in the orthonormal transformation,

$$\|Y\|^2 = Y^T Y = X^T \Phi \Phi^T X = X^T X = \|X\|^2. \quad (5.11)$$

Also, the structure of the data is preserved.

3. The orthonormal transformation diagonalizes the covariance matrix.

From the above discussion, it is easy to see that the discrete K-L expansion only considers the distribution of all samples in the training set, and does not take into account the discrimination between classes. It is necessary to consider the between-class covariance to improve this technique.

5.4 Fisher's Discriminant Method

In his original article [Fis36], Fisher suggested a method of developing canonical variates based on the between-class and within-class covariance matrices.

Let us define the *within-class scatter matrix* of samples to be

$$S_w = \sum_{i=1}^k P(\omega_i) E[(X - M_i)(X - M_i)^T | \omega_i] \quad (5.12)$$

where k indicates the number of classes and M_i is the mean vector of class i . The term ω_i represents the event that X is in class i . And the *between-class scatter matrix* is defined to be

$$S_b = \sum_{i=1}^k P(\omega_i) (M_i - M_0)(M_i - M_0)^T \quad (5.13)$$

where M_0 is the mean vector of all samples.

The feature extraction process is intended to find an $m \times n$ linear transformation matrix A such that only m ($m < n$) feature components $Y = (y_1, y_2, \dots, y_m)^T$ are selected in order to reduce dimensionality, that is,

$$Y = AX \quad (5.14)$$

Also of greatest importance is the enhancement of the similarities within any one class and the differences between classes. In other words, the transformation should be able to make a smaller within-class scatter matrix and a larger between-class scatter matrix. Thus the criterion can be formulated to maximize the divergent metric subject to the transformation matrix A :

$$J = \text{tr}(\Sigma_w^{-1}\Sigma_b). \quad (5.15)$$

The term Σ_w and Σ_b denoting the scatter matrices of Y are

$$\Sigma_w = AS_wA^T \quad (5.16)$$

and

$$\Sigma_b = AS_bA^T. \quad (5.17)$$

In order to maximize J , let us consider

$$\frac{\partial J}{\partial A} = \frac{2S_bA^T(AS_wA^T) - 2S_wA^T(AS_bA^T)}{(AS_wA^T)^2} = 0 \quad (5.18)$$

To satisfy the above equation, we can get

$$S_bA^T - \lambda S_wA^T = 0. \quad (5.19)$$

Thus J is maximized when the rows of matrix A are constructed by the first m eigenvectors of matrix $S_w^{-1}S_b$ [Fuk72]. That is,

$$A^T = (\Phi_1\Phi_2 \cdots \Phi_m) \quad (5.20)$$

where $S_w^{-1}S_b\Phi_i = \lambda_i\Phi_i$ for $i = 1, 2, \dots, n$ and λ_i are ordered as

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n. \quad (5.21)$$

Appendix B presents a technique to solve the eigensystem of matrix $S_w^{-1}S_b$.

Since the distribution of the random vector, X , is unknown, the scatter matrices (also covariance matrices) can be estimated from system response vectors of the sampled patterns in the pattern designation step. Using a standard eigenvector system software package, the feature extraction matrix A can be obtained without difficulty, though it is computationally intensive for high dimensional response vectors. Equation 5.10 describes the criterion for selecting feature components. Suppose that only m feature components are selected out of n components to form a feature vector. The feature extraction matrix thus becomes an $m \times n$ matrix, which can be implemented by a single-layer neural network with linear perceptrons, as shown in Figure 5.1. In summary, the procedure of constructing a

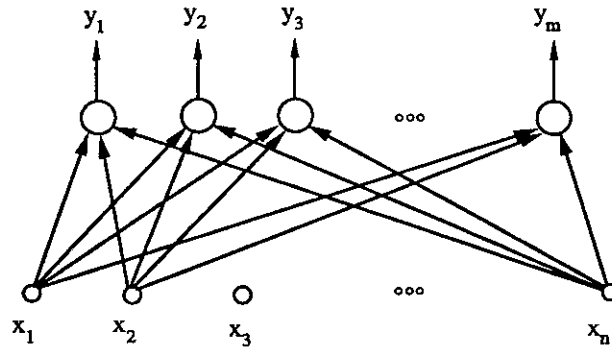


Figure 5.1: A neural network implementation with perceptrons of the feature extractor.

feature extractor is:

1. Estimate the sample scatter (covariance) matrices.

$$S_w = \frac{1}{N - k} \sum_{i=1}^k \sum_{j=1}^{N_i} (X_j^i - \hat{M}_i)(X_j^i - \hat{M}_i)^T \quad (5.22)$$

$$S_b = \frac{1}{k-1} \sum_{i=1}^k \frac{N_i}{N} (\hat{M}_i - \hat{M}_0)(\hat{M}_i - \hat{M}_0)^T \quad (5.23)$$

where $N = \sum_{i=1}^k N_i$, and

$$\hat{M}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} X_j^i \quad (5.24)$$

and

$$\hat{M}_0 = \frac{1}{\sum_{i=1}^k N_i} \sum_{i=1}^k \sum_{j=1}^{N_i} X_j^i \quad (5.25)$$

are the sample mean response vector of class i and the sample mean response vector of all sample patterns, respectively. In addition, X_j^i denotes the j^{th} sample response vector in class i , N_i is the number of samples in class i and k is the number of classes in the parameter space.

2. Calculate the eigenvectors and eigenvalues of the matrix $S_w^{-1}S_b$, normalize the eigenvectors, and sort the eigenvalues in a descending order.
3. Select the first m eigenvectors to construct the transformation matrix A .
4. The feature vector of an observation X is obtained by $Y = AX$ or by standardizing with respect to \hat{M}_0 to yield $Y = A(X - \hat{M}_0)$.

The major properties of this feature extraction scheme are listed:

1. Learning techniques are not used in feature extraction.
2. Little prior knowledge of the problem domain is needed.
3. The computer software package for extracting features is well developed.
4. The underlying mathematical framework is well established.
5. The method is simple and general, but limited to linear transformation. In order to obtain optimal features, patterns are assumed to have a Gaussian

distribution within any one class and identical variance for all classes [Lac75, Fuk90]. In practical applications, these assumptions are often not true.

6. It is very sensitive to noise, discussed in section 5.5.

5.5 An Improvement on the Fisher's Discriminant Method

From Equation (B.7) in Appendix B, the linear transformation of the Fisher's discriminant method scales the system response space by a factor of $\frac{1}{\sqrt{\lambda_i}}$ along the direction of the corresponding eigenvector Φ_i of the within class covariance matrix for all eigenvectors. In fact, the square root of eigenvalue, $\sqrt{\lambda_i}$, is the deviation of the within class scattering along the Φ_i direction. A small eigenvalue means that system responses within any one class are similar along the corresponding eigenvector direction. However, small noise which is not correlated to the within class distribution gets magnified significantly under this transformation. Thus the Fisher's discriminant method performs very poorly with very small noise.

A noise term is introduced to the within class covariance in order to reduce the noise sensitivity of the Fisher's discriminant method. The within class covariance used in computing the eigensystem is then replaced by

$$S_w + S_n, \tag{5.26}$$

where S_n is the autocovariance matrix of noise. The selection of S_n depends on the applications and can be determined from the typical Signal to Noise Ratio (SNR) of the applications and a priori knowledge of noise. If the noise is assumed to be independently identically Gaussian distributed with zero mean and independent

of the system response, the autocovariance of noise becomes a diagonal matrix

$$S_n = \begin{pmatrix} \sigma^2 & 0 & \cdots & 0 \\ 0 & \sigma^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \sigma^2 \end{pmatrix} \quad (5.27)$$

where σ is the standard deviation of the Gaussian noise. Since the noise is assumed to be unbiased, the between class covariance matrix remains unchanged with its autocovariance of noise equal to zero.

There are three implications on the noise term S_n associated with the within class covariance matrix:

1. Computational stability: S_n enlarges the eigenvalues of the within class covariance matrix and stabilizes the computation of inverting the noisy within class covariance matrix. Equation (B.7) also shows the need of the noise term for $\Lambda^{-\frac{1}{2}}$ if very small eigenvalues exist for S_w .
2. Training with noise: S_n can be considered as the additive random noise to the system responses. The feature extractor is thus constructed with a training set presented in a noisy environment.
3. Robustness to noise: S_n reduces the sensitivity of the Fisher's discriminant method to noise. If a priori knowledge about noise is available, S_n is certainly necessary.

5.6 Some Other Analytical Techniques

Since the Fisher's discriminant method is limited to linear transformations, nonlinear analytical techniques may be required to extract nonlinear features and to realize a preprocessor. However, these techniques could be very *ad hoc* and require prior knowledge as well as human expertise. Thus *trial and error* becomes

inevitable in the *ad hoc* approach, especially, when little prior knowledge is available on the applications, and feature extraction performance is unknown in advance. Different analytical techniques are just worth investigating. Two analytical techniques worthy of comment are:

1. frequency domain analysis,
2. exponential components analysis.

The most valuable tool in frequency domain analysis is the Fourier transform which has long been a well known principal analytical technique and extensively applied in signal processing, image processing, and theoretical analysis. Fourier series is a useful representation of signals in the frequency domain, especially for periodic signals. It was not until the invention of Fast Fourier Transform (FFT) [Bri74] that discrete Fourier transforms could be practically utilized on digital computers and were widely applied. The coherence function of power spectra provides a measure of the match between two patterns in the frequency domain [OE78, SK86].

The analysis of exponential component data is frequently encountered in such diverse areas as biology, physics, electrical engineering, and even economics. The amplitudes and decays of the multiple exponential components are thus very promising features for the analysis. Smith et al. [SCS76] and Provencher [Pro76] provide analytical techniques based on the Gardner Transform and utilizing the Discrete Fourier Transform in order to accomplish numerical computation. In a simple case, the multicomponent exponential decays can be expressed as

$$x(t) = \sum_{i=1}^n a_i e^{-\gamma_i t}. \quad (5.28)$$

These methods compute a spectrum of exponential decays and try to extract the information carried by the parameters n , a_i , and γ_i . However, these methods are

1. computationally intensive,
2. of low resolution, and
3. very sensitive to noise as well as rounding errors.

5.7 Summary

In this study, the feature extraction scheme employs the Fisher's discriminant method that maximizes the scattering between pattern classes while minimizing the scattering within any one pattern class. This method also reduces the dimensionality of the pattern vectors. But it is restricted to linear transformation. In addition, *ad hoc* approaches may serve to extract nonlinear features. Moreover, the frequency domain and exponential component analysis techniques might serve as preprocessing tools for the Fisher's discriminant method, depending on the application.

CHAPTER 6

Classification

6.1 Introduction

Classification, which assigns an object to one of several predetermined classes, is fundamental to scientific inquiry and very important in many areas of science and technology. It is considered to be a decision making process which uses either a set of predetermined rules or rules extracted from a training set. The mechanism that performs the classification task is called a classifier. Mathematically, a classifier is a membership function.

Neural network classifiers possess the capability of learning and adaptation which makes a major difference to the conventional classifiers. This chapter begins with a brief description of conventional classifiers. Then a new adaptive neural network classifier is addressed. Also, a two dimensional classification problem is presented to demonstrate the performance of the neural network classifier.

6.2 Conventional Classifiers

There are two principal approaches to the pattern recognition field: statistical pattern recognition and syntactic pattern recognition. The classifiers employed in these two approaches are substantially different. The statistical approach uses a training set to estimate the mathematical model characterizing the patterns, while the syntactic approach uses language grammar provided by experts.

6.2.1 The Classifiers of Statistical Pattern Recognition

In statistical pattern recognition, classification may be accomplished with classifiers with parametric techniques or with classifiers with nonparametric techniques.

A parametric technique requires knowledge of the structure of the underlying probability distribution of the patterns. Otherwise, a strong assumption must be made about the distribution. A nonparametric technique, on the other hand, does not require knowledge of the distribution. Some of the nonparametric techniques only attempt to estimate the distribution. The nonparametric techniques are important because the knowledge is usually not available a priori or the assumption of underlying probability distribution is not justified. In pattern recognition applications, this information is generally not obtainable. Therefore the parametric classification scheme is of little use.

Some popular nonparametric classification schemes are [DH73]:

1. linear classifiers: The discriminant function is a weighted sum of the feature components. The decision boundary is a hyperplane.
2. nonlinear classifiers: The discriminant function is a nonlinear combination of the feature components. The quadratic classifier is an example.
3. piecewise classifiers: For the multiclass problem, the decision boundary must have a piecewise structure. In general, the piecewise function is linear or quadratic.
4. minimum distance classifiers: For each class, there is one representative prototype pattern. The testing pattern is compared with all the prototype patterns to determine which one it is closest to.

5. nearest neighbor: Basically, this kind of classifier is similar to minimum distance classifiers except that the testing pattern is compared with a set of sample patterns of known classification. There may be more than one sample pattern in each class.
6. k -nearest neighbor: This scheme determines the k nearest prototype patterns to the testing pattern, and uses the *majority* of equal classification as the classification of the testing pattern.

6.2.2 The Classifiers of Syntactic Pattern Recognition

The syntactic pattern recognition [Fu82] approach is based on the utilization of concepts from formal language theory. It employs syntactic grammars to describe the structure and interrelationships between the primitive components of a pattern. The extraction of the primitive components from a pattern requires a priori knowledge. No general training algorithms have yet been discovered. Some approaches of identifying the primitive components rely on statistical pattern recognition techniques. In general, context free grammar is used in the syntactic approaches, and the classifier of a syntactic system is a push down automata.

6.3 Neural Network Classifiers

In this study, we propose an adaptive neural network as the pattern classifier. This neural network classifier is based on the architecture of the Hamming net and capable of adapting to higher classification performance. Because of the lack of prior knowledge about the system responses and the difficulty of describing the output signals of a system in a formal language, syntactic approach is not employed.

6.3.1 The Neural Network Classifier Based on the Hamming Net

The classifier for the Fisher's feature extraction scheme described in section 5.4 can be designed in the sense of Euclidean distance to find the nearest exemplar representing a pattern class. That is,

$$\min_i \{(Y - AM_i)^T (Y - AM_i)\} \quad (6.1)$$

where Y is the feature vector obtained from the output of the feature extractor, A is the transformation matrix, and M_i which is the mean pattern vector of pattern class i is designated to be an exemplar. In fact, this is a *minimum distance classifier* with the mean pattern vectors as exemplars. This expression is equivalent to

$$\max_i \{(Y - \frac{AM_i}{2})^T AM_i\}. \quad (6.2)$$

This classifier can be implemented with an artificial neural network of architecture similar to the Hamming net [Lip87] except that this classifier deals with real valued vectors instead of binary vectors. A two-layer neural network is constructed in such a way that the first layer calculates the similarities of the feature vector Y to all the stored feature exemplar AM_i 's, and the second layer selects the maximum. Thus, the feature vector Y is classified with the pattern class of the most similar feature exemplar.

Figure 6.1 shows a neural network implementation of the classifier. The perceptron of the first layer computes the following function:

$$u_i = \sum_{j=1}^m w_{ij} y_j - (AM_i)^T AM_i / 2 \quad (6.3)$$

where m is the number of feature components, and u_i is the similarity or likelihood of Y to AM_i . In this equation, the connection weight is defined to be

$$w_{ij} = \Phi_j^T M_i \quad (6.4)$$

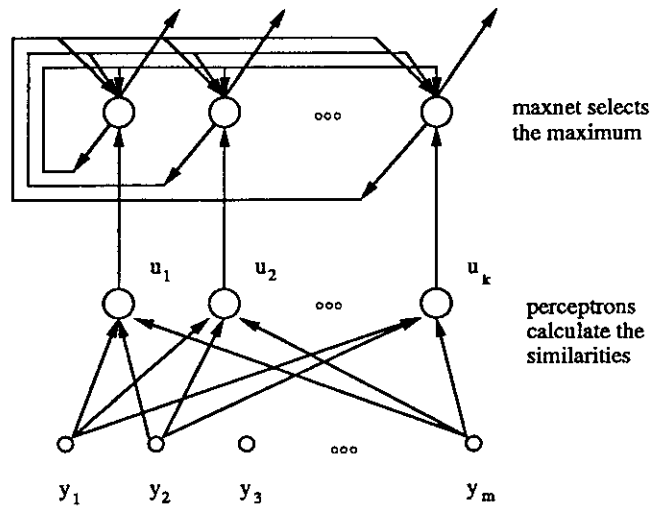


Figure 6.1: A neural network classifier.

where Φ_j is the j -th row of matrix A .

The second layer is a maxnet which performs the winner-take-all operation, described in section 3.4.5. Its input is fed from the output of the first layer prior to time zero and then removed. After convergence, the node with a positive output value indicates the class which the feature vector Y is classified with.

The decision boundary constructed from the minimum distance property of the exemplars is actually a Voronoi diagram [Sed90] of these exemplars. The Fisher's discriminants utilizing a minimum distance classifier perform optimal classification if every class is Gaussian distributed and has identical variance [Lac75]. In applications, these conditions generally do not exist. Hence, two problems arise:

1. The exemplar of every class is not necessarily the mean pattern vector because the sizes of the class regions in the pattern space are not identical. In addition, patterns of a class may scatter asymmetrically with respect to their mean. The assignment of exemplars becomes an important issue.
2. The shape of the region of a class may be concave. Thus, one exemplar for

each class may not be enough to cover the nonconvex region.

6.3.2 Modified LVQ

An adaptive nearest neighbor classifier with multiple exemplars per class is proposed to cope with these two problems. A learning technique is employed to adapt the exemplars using the training set. This classifier is a modification of Kohonen's LVQ [Koh84].

In a pattern recognition system, the inputs to a classifier are feature vectors. But for the sake of notational consistency, patterns are directly used as the inputs to the classifier in the remaining discussions of this chapter.

Based upon the nearest neighbor classification, the learning rule of LVQ is as follows:

1. *Punishing*: if a training pattern X is misclassified with the exemplar M_l , the exemplar is pushed away from X .

$$M_l(t+1) \longleftarrow M_l(t) - \eta(X - M_l(t)). \quad (6.5)$$

2. *Rewarding*: if X is classified with M_l correctly, the exemplar is pulled toward X .

$$M_l(t+1) \longleftarrow M_l(t) + \eta(X - M_l(t)). \quad (6.6)$$

In the above two equations, t is the learning time in a discrete form. And $0 \leq \eta < 1$ is the learning rate which decreases monotonically with time. The convergence of this algorithm is guaranteed as η goes to zero.

After the learning process of LVQ converges, the exemplars of a class get distributed approximately with a distribution of the training patterns of this class [Koh84]. Every exemplar of a class dominates a region with its size approximately

inversely proportional to the training pattern density of this class. In other words, the number of training patterns classified with an exemplar using nearest neighbor classification scheme is about the same for all exemplars of a class. Let us define the *utilization* of an exemplar as:

$$U(M_l) = \frac{\text{Number of patterns in class } k \text{ classified with } M_l}{\text{Total number of patterns in class } k}, \quad (6.7)$$

where the exemplar M_l is one of the representative patterns of class k . If the training patterns are sampled uniformly from the pattern space, every exemplar of a class is of equal importance and similar utilization.

However, LVQ cannot define the correct decision boundary clearly. It always misclassifies patterns near the decision boundary because the exemplars are repelled by the training patterns of other classes according to the *punishing* rule and attracted inwards to the region of their representing classes by the *rewarding* rule. Each exemplar is located deeply inside its class region and not able to define the decision boundary accurately. Hence, a Modified LVQ, or MLVQ, is proposed to remedy the boundary problem, while maintaining the utilization of exemplars.

The learning rule of the MLVQ is:

1. For correct classification of a training pattern X with exemplar M_s , the learning process only takes place in the early several training epochs and

$$M_s(t+1) \leftarrow M_s(t) + \eta(X - M_s(t)). \quad (6.8)$$

2. If X is misclassified with M_l , and the nearest exemplar of the correct class to X is M_s ,

- (a) punish M_l

$$M_l(t+1) \leftarrow M_l(t) - \eta(X - M_l(t)) \quad (6.9)$$

and

(b) reward M_s

$$M_s(t+1) \leftarrow M_s(t) + \eta(X - M_s(t)). \quad (6.10)$$

Here, one training epoch is defined to be the application of the learning rule by going through every training pattern once. Step 1 of the learning rule distributes the exemplars inside the regions of their classes so that they have similar utilization. After spreading the exemplars over their corresponding regions, only misclassification is taken to perform the learning process and step 1 is skipped from then on. The time to terminate learning from correct classifications can be recognized from the misclassification difference between two consecutive training epochs. In experiments, this task is usually done within 10 training epochs. This number depends on the number of exemplars, the initial values of the exemplars, the size of the training set, and the learning rate η . For a larger number of exemplars, more training epochs may be required to settle down the competition among exemplars. Step 2 of the learning rule then refines the decision boundary by punishing the exemplar of the wrong class and rewarding the exemplar of the correct class whenever misclassification occurs.

The remaining problems are how to assign the initial values of the exemplars and how many exemplars of a class suffice. In order to speed up the learning process of step 1 of the MLVQ learning rule, the initial values of the exemplars of any class are set to be the mean vector of this class with small additive noise. If the region of the class is convex, the exemplars are already inside this region. Otherwise, these exemplars are none-the-less very close this region.

The number of exemplars of a class is determined in an iteratively incremental manner. After the learning rule converges, if misclassification occurs in some classes, more exemplars are added for these classes and the learning process is repeated. However, if the misclassification rate cannot be reduced after adding more

exemplars, region overlaps between different classes are inherent in the problem.

In summary, the algorithm for constructing an adaptive neural network classifier is:

Algorithm: Training of the MLVQ Classifier

1. Select the initial number of exemplars for each class.
2. Set the initial values of the exemplars to be the mean vectors of their representing classes with small additive noise.
3. Choose the initial learning rate η .
4. For all training patterns, apply the MLVQ learning rule to adapt the exemplars.
5. If all training patterns are classified correctly, stop.
6. If any misclassification occurs and η is greater than zero, decrease η and go to step 4.
7. If η has been reduced to zero, assign more exemplars to where the misclassifications take place in the pattern space and go to step 3.
8. If the misclassification rate cannot be improved after executing step 7, stop.

In step 7, the new exemplars can be assigned to locations where the misclassifications take place, or the method mentioned in step 2 can be used. After the algorithm stops, each exemplar is assigned to one perceptron whose connection weights are the vector components of this exemplar. The neural network implementation of the MLVQ classifier is shown in Figure 6.1.

6.4 Example

In the following, the MLVQ is applied to a two dimensional classification problem. A training set of 5000 patterns and a testing set of 5000 patterns are sampled randomly from the pattern space. Figure 6.2 shows the decision boundary of three classes and the initial locations of 21 exemplars using MLVQ. Figure 6.3 shows the final locations of these exemplars with almost perfect classification for both the training set and the testing set. These exemplars are then used to construct the classifier for the classification problem.

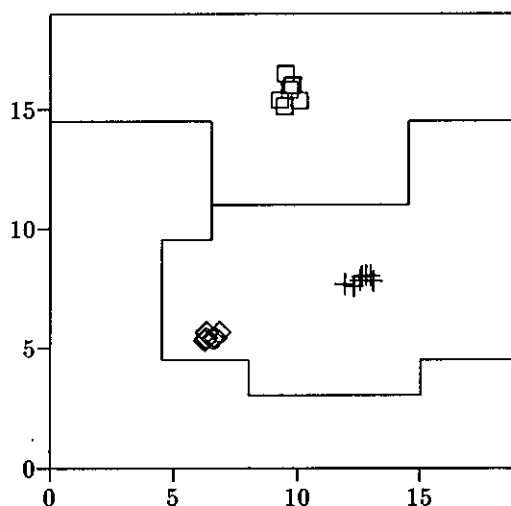


Figure 6.2: Three pattern classes and the initial locations of exemplars for the three classes using MLVQ.

In the experiment, step 1 of the MLVQ learning rule stopped learning after 10 training epochs. The initial η was set to be 0.05 and linearly reduced to 0 at training epochs 400. The learning process was terminated after 400 training epochs if a perfect classification of the training set was not obtained.

The effectiveness of training is depicted in Figure 6.4 where the number of

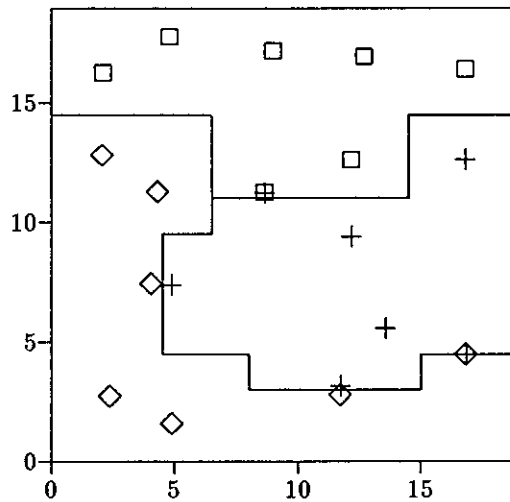


Figure 6.3: The final locations of the exemplars using MLVQ.

misclassification is plotted against the training epochs and the number of exemplars per class using the MLVQ and LVQ. It is easy to see that MLVQ outperforms LVQ both in learning speed and in misclassification rate. LVQ does not obtain perfect classification during training and seems to be learning saturated after 50 training epochs even with a large number of exemplars.

Table 6.1 presents the misclassification rates of the testing set for different methods with various number of exemplars per class. ADSM [GS91] stands for the adaptive decision surface mapping whose learning rule is identical to the step 2 of the learning rule of MLVQ. MLVQI indicates the method of MLVQ with *intensive learning* scheme, described below. The results show that MLVQ produce a better performance than ADSM and both of them outperform LVQ.

The order of presenting the training patterns affects the learning result. Let us define *intensive learning* as a learning process by presenting training patterns of the same class consecutively together and *random learning* by randomly presenting training patterns regardless of their class. In general, intensive learning performs

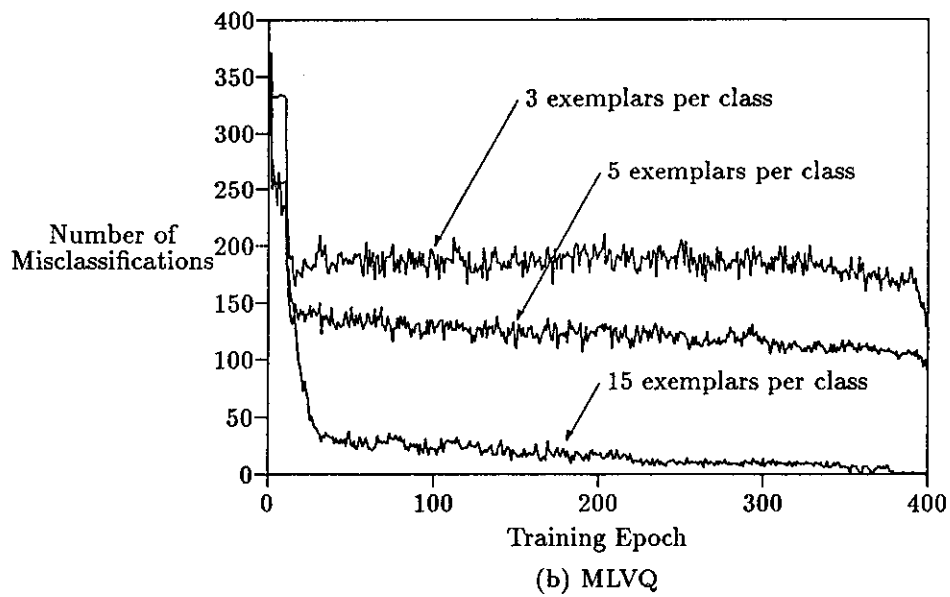
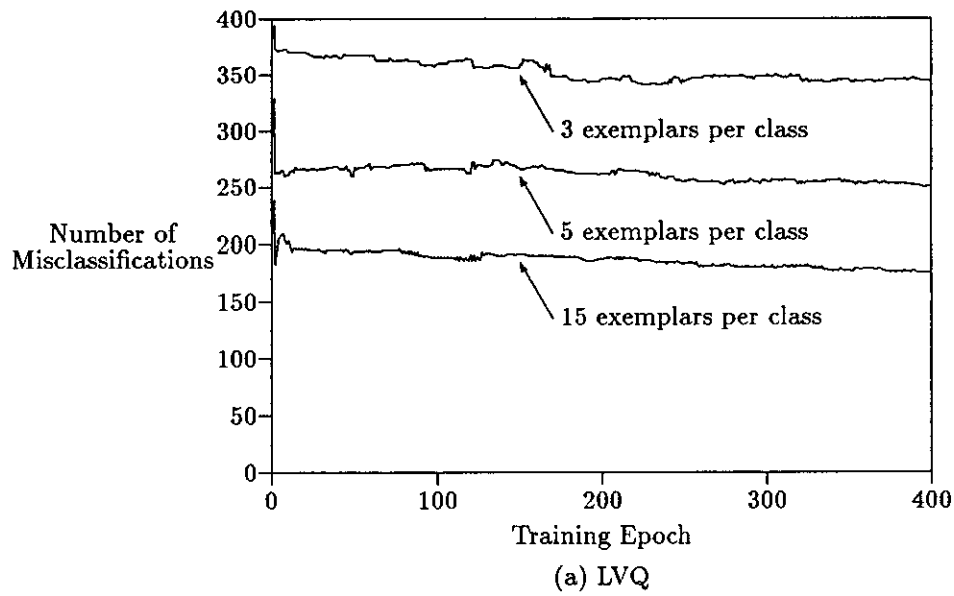


Figure 6.4: The number of misclassifications in the training history using LVQ and MLVQ with 3, 5, and 15 exemplars per class.

Table 6.1: Misclassification rates (%) on the testing set using different methods.

number of exemplars per class	MLVQ	MLVQI	ADSM	LVQ
1	29.34	39.34	27.96	24.78
2	9.02	28.74	10.20	14.40
3	2.26	3.24	6.48	6.84
4	1.58	6.02	1.60	6.26
5	1.34	2.20	1.60	5.02
6	0.52	0.54	0.70	3.80
7	0.36	0.52	1.10	2.70
8	0.70	0.22	3.12	3.42
9	0.64	0.28	1.48	3.50
10	0.10	0.24	0.60	3.18
15	0.02	0.30	0.62	3.52
20	0.14	0.12	0.38	2.22
25	0.26	0.08	0.58	2.22

slightly better than random learning in terms of learning speed, especially in the early learning stage, as can be seen in Figure 6.5. However, when the number of exemplars is not large enough, intensive learning leads to an overshooting behavior of learning.

Since the initial exemplars of ADSM are randomly drawn from the training set, performance degenerates if the initial exemplars are not spread evenly over the class regions. Figure 6.6 depicts the initial locations of 30 exemplars with slightly biased distribution. In this case, ADSM only obtains a 2% misclassification rate. Also, some exemplars are not utilized and some never learn anything from the training set, as can be seen in Figure 6.7. It is necessary to redistribute these exemplars. Thus, step 1 of MLVQ learning rule helps improving the utilization of exemplars and gives better initial exemplars for ADSM.

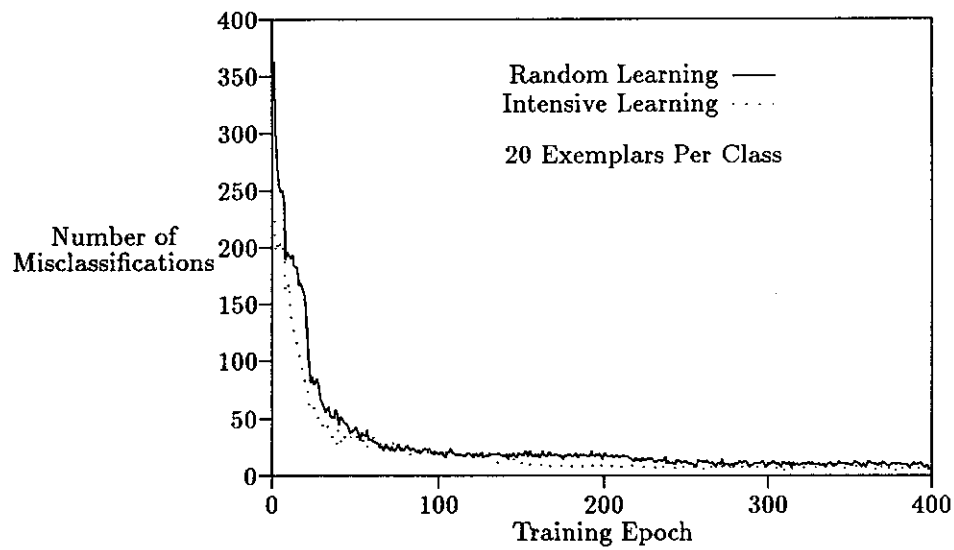
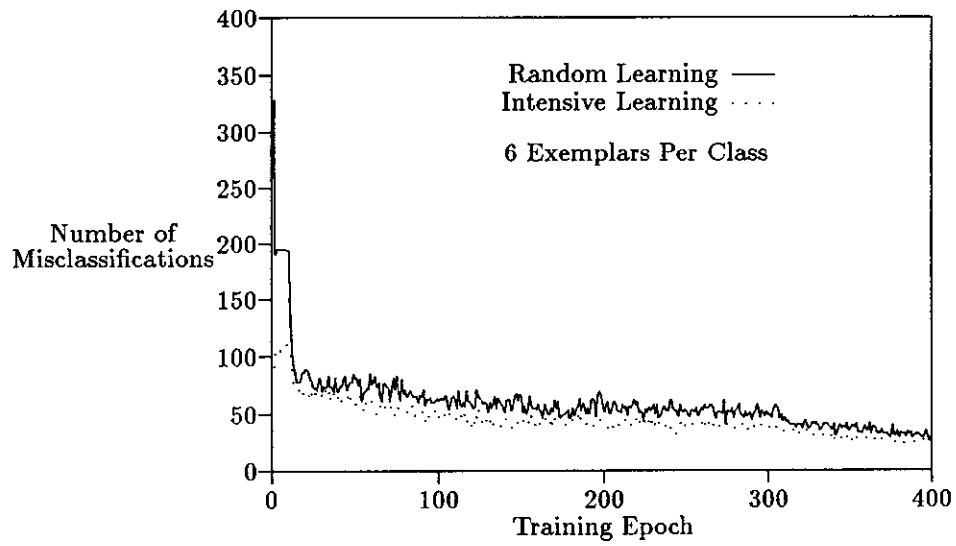


Figure 6.5: Intensive learning vs random learning.

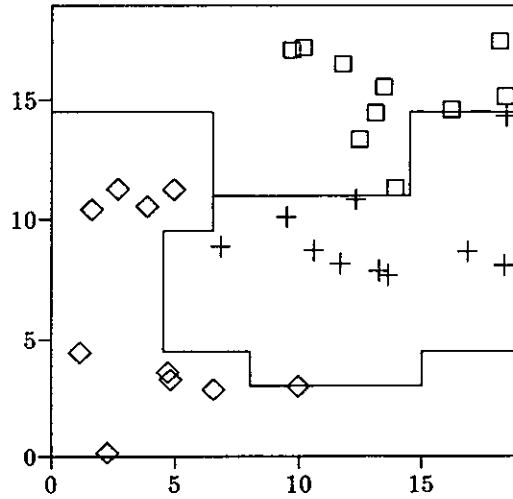


Figure 6.6: The initial locations of the exemplars with biased distribution using ADSM.

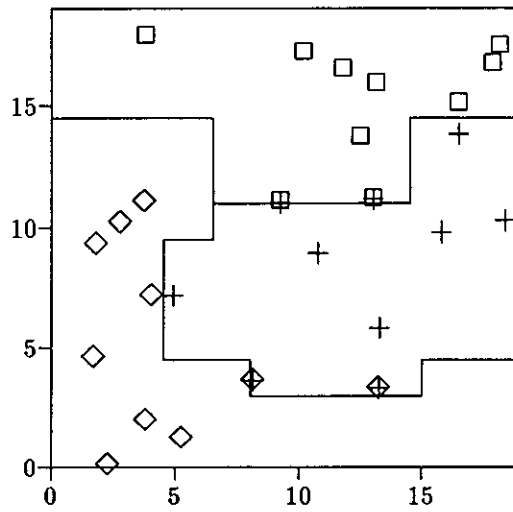


Figure 6.7: The final locations of the exemplars using ADSM.

6.5 Discussion

There are two issues regarding the improvement of the classifier that require further discussion:

1. training with noise,
2. null class classification.

6.5.1 Training with Noise

Training with noise is concerned with presenting the training patterns with additive random noise to the classifier during training stage. There are several reasons for training with noise:

1. The training patterns are only a small fraction of total patterns. Patterns in the neighborhood of a training pattern are probably in the same class as this training pattern. Using the original training set with additive noise thus increases the number of possible training patterns. The selection of its noise amplitude depends on the sparseness of the training patterns in the pattern space.
2. In the real world, patterns are usually measured in a noisy environment. It is practical to train the classifier with noise because the classifier eventually takes the noisy patterns as inputs in the recognition stage. However, the prior information on the random noise is not available.
3. If there are gaps between class regions in the pattern space, noise could expand the territories of pattern classes and improve the classification performance.

On the other hand, noise could also blur the decision boundary, confuse the classifier, and increase the training time.

The two dimensional classification problem is again used as an example to demonstrate the effectiveness of noise on the learning results when sparse training patterns are presented. The training patterns are sampled at the integer grid points (i, j) except those points on the boundary of two classes, where i and j are integer and in the range $[0, 19]$. The testing set with 5000 testing patterns is the same as the one used in the previous example.

The noisy version of a training pattern (i, j) is assumed to be located on a circle with this training pattern as its origin and a given radius r which is defined to be the noise amplitude. Thus the noisy pattern is $(i[r], j[r])$ with

$$i[r] = i + r \cos(\theta), \quad (6.11)$$

$$j[r] = j + r \sin(\theta), \quad (6.12)$$

where $\theta \in [0, 2\pi]$ is a uniformly distributed random variable. Two cases dealing with how the noisy patterns are incorporated with the original training patterns during training stage are investigated. In case I, the noisy patterns are generated before hand and associated with the original training patterns as a training set. Once the noisy patterns are generated, they are fixed and repeatedly used to train the classifier. The number of noisy patterns that should be included for every original training pattern is also a parameter of interest under study. In case II, only one noisy pattern for every original training pattern is incorporated with the original training patterns to train the classifier. But, in each training epoch, new noisy patterns are generated.

Figure 6.8 depicts the misclassification rates against different noise amplitudes using MLVQ with seven exemplars per class and eight additional noisy patterns

for each training pattern in case I. Since noise is random, each data point of the misclassification rate in the plot is an average of four simulation results. Starting with small noise, the classification performance is improved gradually when noise amplitude gets larger. The best classification performance can be achieved if the noise amplitude is approximately 0.5. Beyond that point, the performance degenerates dramatically as the noise amplitude increases. It is obvious that the overlaps between noisy class regions occur when noise amplitude is greater than 0.5.

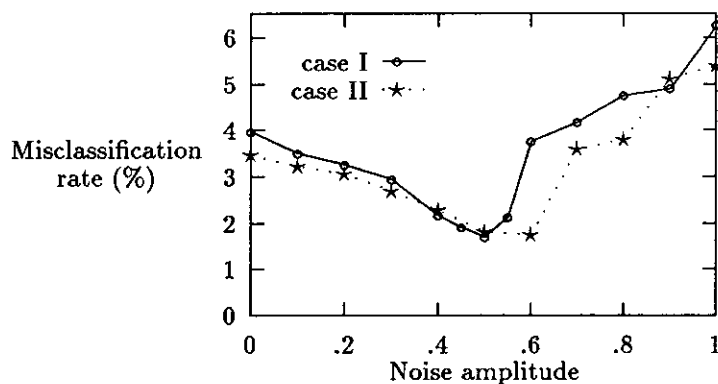


Figure 6.8: Misclassification rates (%) of MLVQ against noise levels under training with noise scheme.

Figure 6.8 also shows that case II is more effective in using noisy patterns than case I. Case II requires not only less memory space to store the whole training set, but also less training time. In addition, case II outperforms case I in terms of the misclassification rates. Since new noisy patterns are required in each training epoch, the case II approach is certainly better than case I if it is easy to generate a noisy pattern from an original pattern.

For case I, the misclassification rates plotted against the number of noisy patterns for each original training pattern with noise amplitude 0.5 and seven exem-

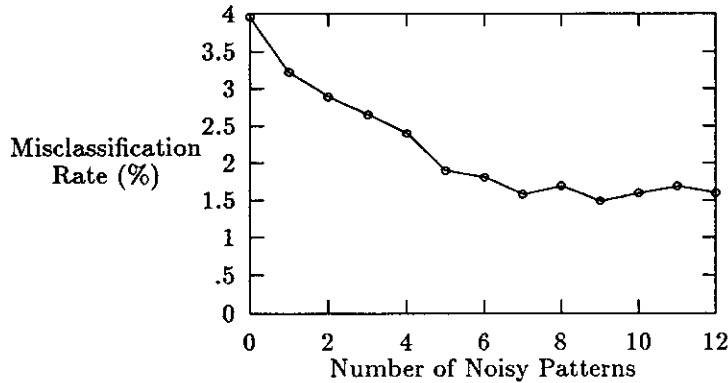


Figure 6.9: Misclassification rates (%) against the number of noisy patterns per original training pattern used in training with noise scheme.

plars per class are depicted in Figure 6.9. Even with an appropriate noise amplitude, the performance is not improved linearly with the number of additional noisy patterns though the training time and memory space required increase linearly with the number of noisy patterns. In the two dimensional pattern space, no obvious performance improvement can be obtained when the number of noisy patterns for every original training pattern is greater than 6.

6.5.2 Null Class Classification

For a pattern located outside the pattern space of an application, this pattern should not be classified with any of the designated pattern classes. It is necessary to designate a null class for those outsiders.

The perceptron implementation of Equation (6.2) does not provide any bound to exclude outsiders. For instance, if $Y = bAM_i$ and b is a positive scalar, the similarity function of Equation (6.3) cannot be bounded from above. Moreover, for any feature vectors orthogonal to AM_i , there is no information to distinguish the distances from these feature vectors to AM_i by the similarity function.

Since the training set does not include the outsiders, every exemplar requires a

similarity or distance bound in order to detect the outsiders. The first order computation with the perceptron implementation is obvious not suitable for applications requiring null class classification. Nevertheless, the second order computation of Equation (6.1) provides more information on setting bounds to exclude the outsiders. It is for sure that the lower bound of the Euclidean distance to an exemplar is zero. In this case, the pattern is identical to the exemplar. The upper bound can be obtained using the training set. Every exemplar has its own Euclidean distance bound to detect the outsiders. Given a pattern, if all exemplars identify it as an outsider, this pattern is classified into the null class.

6.6 Conclusions

MLVQ uses both misclassification and correct classification of training patterns to adapt the exemplars during the early learning stage. When the training outcome is about to get saturated, only misclassification is taken into consideration in correcting the decision boundary. In this manner, MLVQ not only defines an accurate decision boundary, but also maintains similar utilization of exemplars for the purpose of fault tolerant implementation. The results show that MLVQ is better than ADSM and LVQ in terms of misclassification rates. In this study, the MLVQ classifier is applied to the pattern recognition system for nonlinear parameter characterization.

The issue on training with noise is also discussed. When the training patterns are sampled sparsely in the pattern space, small noise associated with the training patterns helps to improve the classification performance. It is found that presenting new noisy patterns in each training epoch is a good approach to reducing the required memory space and training time in comparison to presenting fixed noisy patterns. The noise amplitude for improving performance depends on the sparse-

ness of the training patterns and the gaps between class regions in the pattern space.

CHAPTER 7

Performance Evaluation

Performance evaluation not only provides an indication of the confidence level of the classification result, but also evaluates the problem solving process from pattern designation, to feature extraction, to pattern classification. When a pattern is classified, it is very important to know how reliable the classification result is. Similarly, the performance of the pattern recognition approach is evaluated in order to remedy any drawback and imperfection. The quality of both tasks is a measurement based on the estimation of classification performance. If the classification result under evaluation is not satisfactory, every step of the problem solving process should be analyzed judiciously to reexamine its correlation to the problem nature. For instance, some *ad hoc* approaches may be applied to nonlinear feature extraction in order to improve the performance. The correct classification rate provides a measure of confidence in the realization of the mathematical descriptors of the unknown parameters.

7.1 Testing Set Generation

Because the patterns in the pattern vector space are often uncountable or impossible to enumerate exhaustively, a subset of all patterns called the “testing set” is required to estimate the classification performance. The testing set which consists of patterns and their class identities may include:

1. *the training set*: This set can only evaluate the effectiveness of training the classifier. It would underestimate the misclassification rate of the patterns

in the whole pattern vector space. However, if the training set is sampled uniformly from the pattern vector space and is large enough, the estimation of classification performance can be generalized to the whole space.

2. *patterns with the same function approximation as that of the training set but generated randomly.* This kind of testing set is often used if patterns are easy to generate and store.
3. *patterns with a different function approximation to that of the training set.* This testing set is to generalize the estimation result to the problem that is not limited to certain parameterized models.
4. *the above three sets corrupted by different levels of noise.* This is to estimate the effect of noise on the classifications.

7.2 Estimating the Confidence Level of Classification

After performing classification on the testing set, a classification result matrix $R = [r_{ij}]$ can be obtained, where r_{ij} is the number of patterns in class i classified with class j . Assume that a cost function $\lambda(p_a)$ for any pattern p_a is available. The cost of a pattern is counted when it is assigned to a correct class. Thus the overall correct classification rate is

$$\frac{\sum_{i=1}^k \sum_{\{p_a | p_a \in C_i \ \& \ p_a \mapsto C_i\}} \lambda(p_a)}{\sum_{\{p_a | p_a \in C\}} \lambda(p_a)} \quad (7.1)$$

where k is the number of pattern classes, C is the testing set, and $p_a \mapsto C_i$ denotes that p_a is classified with class i . If all patterns are equally costly, the overall correct classification rate becomes

$$\frac{\sum_{i=1}^k r_{ii}}{\sum_{i=1}^k \sum_{j=1}^k r_{ij}}. \quad (7.2)$$

Of special interest here is to estimate the reliability of a classification result. Let us define the measure of the quality to be the probability that a pattern belongs to class i when it is classified with class i ; that is the probability

$$P[p_a \in C_i | p_a \mapsto C_i], \quad (7.3)$$

Applying Bayes rule to replace the a posteriori probability by the a priori probabilities and conditional probabilities, we get

$$P[p_a \in C_i | p_a \mapsto C_i] = \frac{P[p_a \mapsto C_i | p_a \in C_i] P[p_a \in C_i]}{\sum_{j=1}^k P[p_a \mapsto C_i | p_a \in C_j] P[p_a \in C_j]}. \quad (7.4)$$

If the a priori probability $P[p_a \in C_i]$ is not available, it can be estimated by

$$\frac{\sum_{j=1}^k r_{ij}}{\sum_{i=1}^k \sum_{j=1}^k r_{ij}} \quad (7.5)$$

by assuming that the testing set is sampled uniformly from the pattern vector space. Also $P[p_a \mapsto C_i | p_a \in C_j]$ is estimated by

$$\frac{r_{ji}}{\sum_{i=1}^k r_{ji}}. \quad (7.6)$$

Therefore the a posteriori probability of Equation (7.4) yields

$$\frac{r_{ii}}{\sum_{j=1}^k r_{ji}} \quad (7.7)$$

which provides a measure of confidence on the classification result that the pattern p_a is classified with class i . In addition,

$$P[p_a \in C_i | p_a \mapsto C_j] = \frac{r_{ij}}{\sum_{i=1}^k r_{ij}}. \quad (7.8)$$

7.3 Class Region Growing

Since the regions in the pattern vector space are contiguous and the territories of pattern classes are disjoint and exhaustive, two adjacent pattern classes are

separated by a decision boundary in the pattern vector space. Small noise level can cause misclassification of patterns close to the decision boundary. Figure 7.1 shows a scenario of the fuzzy decision area and the mapping from the parameter space to the pattern vector space.

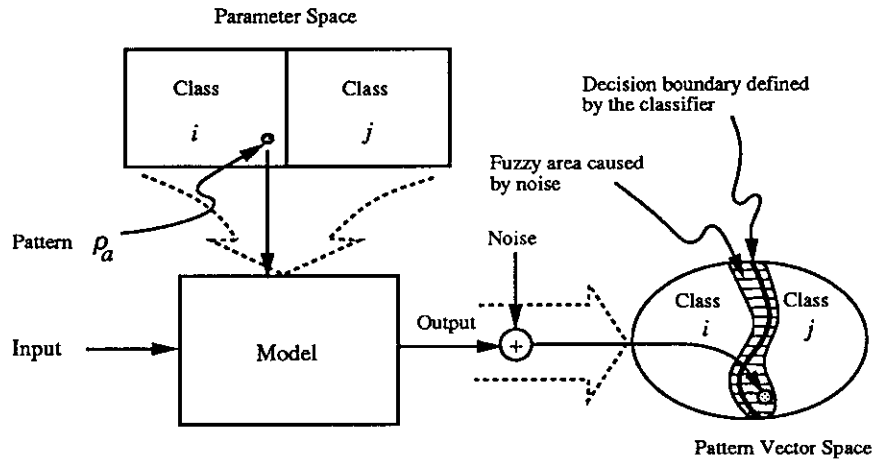


Figure 7.1: The mapping from the parameter space to the pattern vector space.

The fuzzy decision area is caused by noise, nonlinear mapping, and imperfect feature extraction and classifier training. Consider only the effect of noise because the later two effects exist inherently in an application and the problem solving approach. A pattern p_a belongs to class i , while being classified with class j , because of the factors mentioned above. In order to make the classification result useful, the parameter space of class j should contain the pattern p_a . It becomes necessary to extend the parameter space of class j into that of class i , vice versa (see Figure 7.2). There exists an overlap region along the border of class i and class j in the parameter space. Patterns inside this region can be classified with either classes. The identified parameter space still contains the pattern p_a . An optimization search or further refinement partition on this identified parameter space can be conducted to locate the best model for a real world system.

The procedure for growing class regions is:

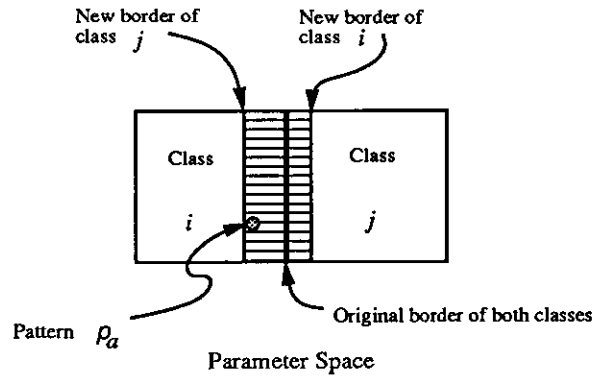


Figure 7.2: The parameter space overlap between class i and class j .

1. Select a testing set with a predetermined level of additive noise.
2. Perform classification on the testing set and mark those patterns which are in class i , but classified with class j .
3. Expand the territory of class j into that of class i along their border in the parameter space in order to include those marked patterns. In the systematic or stratified sampling schemes, a grid network is superimposed on the parameter space. Territory expansion is based on the grid unit containing one marked pattern.
4. Repeat step 2 and 3 for all pairs of adjacent pattern classes.

After the region growing process, a ratio of size of the overlapped regions to the size of the whole parameter space can be calculated. If the ratio is close to one under some noise level, it is meaningless to continue partitioning the parameter space of any class region. Noise effect gets larger when the parameter space becomes smaller. In this case the best model can only be identified within a subregion of the parameter space which is actually an error bound of the best model.

7.4 Class Region Merging

Nonlinear mapping from the parameter space to the pattern vector space may cause class decision region overlap in the pattern vector space for two nonadjacent pattern classes in the parameter space. The overlap ratios of either classes can be estimated from the testing set. The union of these two classes as one class is the best way to make the classification result useful.

7.5 Summary

The approach to analyze the overall classification performance and the reliability of a pattern classification result is discussed. The classification performance is estimated from a testing set. According to the top down dichotomy algorithm, each pattern class has the same number of sample patterns and similar variation in the response space. Thus the pattern vectors distribute fairly evenly in the response space and are very suitable for estimating the classification performance.

The overall classification performance provides an indication of the efficiency of the problem solving process from pattern designation, to feature extraction, to pattern classification. If the classification result is not satisfactory, the problem solving process should be analyzed to improve the performance.

The reliability of a pattern classification is the confidence level associated with this classification result which provides a functional region for further search for the actual function. In fact, a misclassification of a pattern near a boundary is not especially harmful because the solution is still very close the identified region though not inside this region.

Part III

Applications

CHAPTER 8

Case Study I: Characterization of a Nonlinear Torsional Spring

8.1 Introduction

This chapter reports on the application of the proposed methodology to identify the nonlinear characteristic of a torsional spring of a simple articulated joint system. Simulation results are also presented. Both pattern recognition and optimization approaches were applied to the identification of this simple system, in order to provide a basis for comparison.

8.2 The Simple Articulated Joint System

8.2.1 The System

The system to be modeled is shown in Figure 8.1. Two rigid bars of negligible mass are connected by a torsional spring at B. A point mass m is attached to the second bar at C. By applying a torque to the first bar to generate an angular velocity $\omega(t)$, the angular deflection θ of the spring is measured in time.

The dynamic system can be expressed in the following equation.

$$mL^2\ddot{\theta} + m\omega^2 RL \sin \theta + k(\theta)\theta = -mL(L + R \cos \theta)\dot{\omega}. \quad (8.1)$$

For small angle of θ , this equation can be linearized:

$$mL^2\ddot{\theta} + (k(\theta) + m\omega^2 RL)\theta = -mL(L + R)\dot{\omega}. \quad (8.2)$$

Since θ is assumed to be small, Equation (8.2) will be used in the following discussions. Applying a driving torque to the first bar such that $\omega(t)$ behaves in the

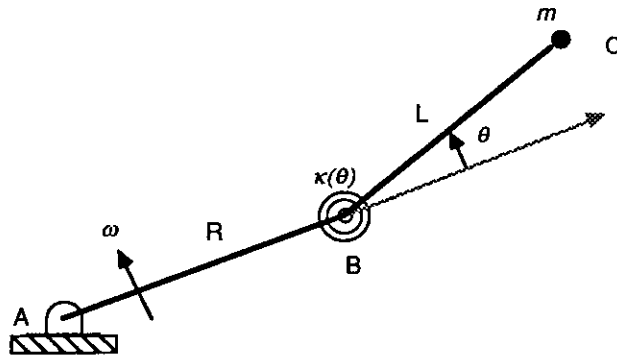


Figure 8.1: The articulated joint system of two rigid bars connected by a torsional spring.

Table 8.1: System parameters.

<i>variable</i>	<i>value</i>	<i>unit</i>
Ω	π	rad./sec
T	30	second
L	2	ft
R	2	ft
m	1	slug

following equation

$$\omega(t) = \begin{cases} \frac{\Omega}{T}(t - \frac{T}{2\pi} \sin(\frac{2\pi t}{T})) & \text{if } t < T \\ \Omega & \text{if } t \geq T \end{cases} \quad (8.3)$$

and given some other system constants in Table 8.1, Figure 8.2 shows the simulation result with linear spring $k(\theta) = 29.61$ and a plot of $\omega(t)$.

8.2.2 The Objective

The objective is to formulate a model that characterizes the nonlinear behavior of the spring, using the observed $\theta(t)$ as the raw data. For simplicity, we will study the behavior of spring response force instead of the Hooke's constant. Let us define

$$f(\theta) = k(\theta) \cdot \theta.$$

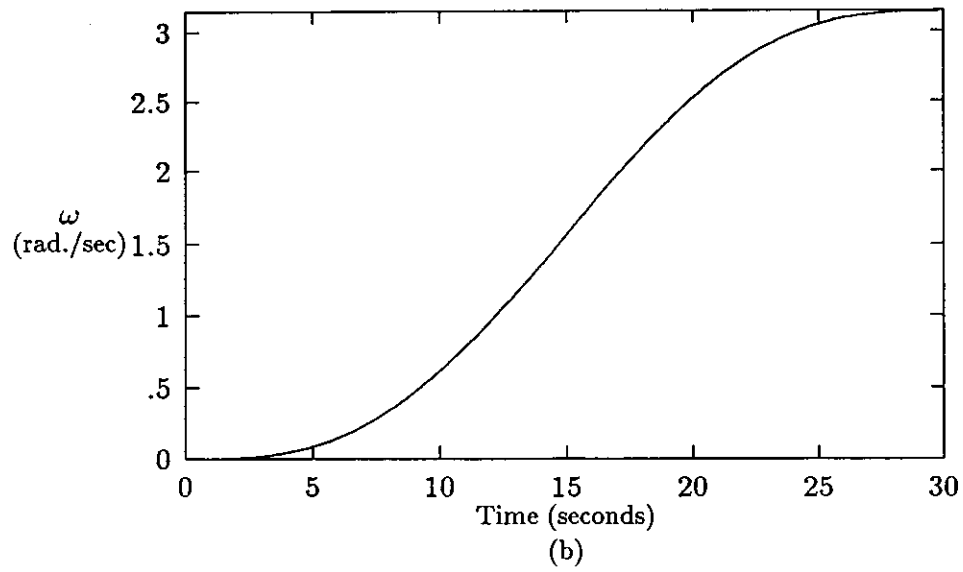
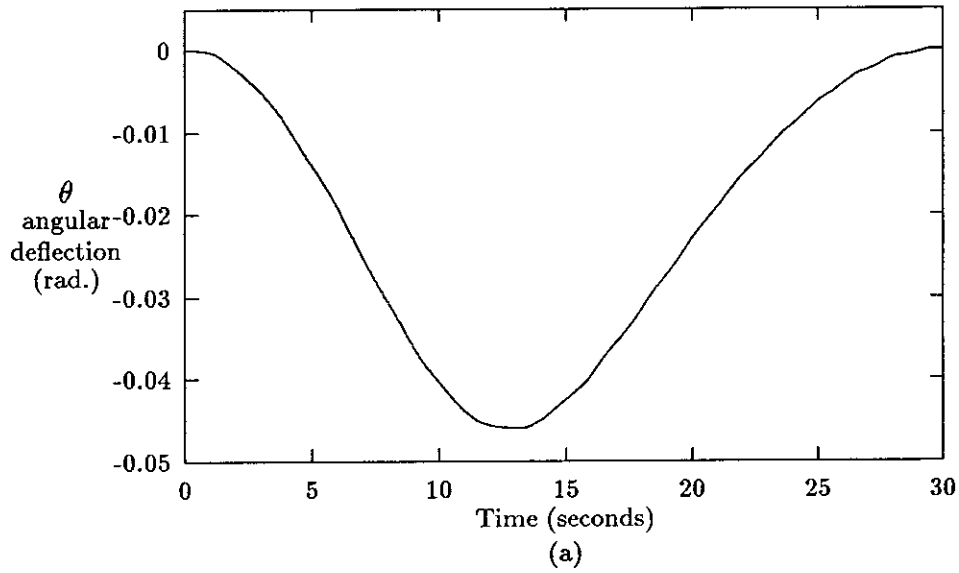


Figure 8.2: (a) A simulation result of system response at $k(\theta) = 29.61$. (b) A plot of $\omega(t)$.

Therefore the objective becomes the realization of the characteristics of the spring torque as a function of the spring's angular deflection.

8.2.3 Classification Results

Define the $n \times 1$ column vector X to be the system response sampled from the angular deflection θ of the unknown spring. The experimental data were obtained analytically by computer simulation. The data are sampled at a sampling period of T/n from time $t = T/n$ to time $t = T$. That is,

$$X = (x_1, x_2, \dots, x_n)^t = (\theta(T/n), \theta(2T/n), \dots, \theta(T))^t,$$

where superscript t denotes the transpose operator; and $n = 256$ in this experiment.

The variables of the parameterized approximation of $f(\theta)$ define the parameter space of the model that possibly models the system. Applying the proposed methodology, the computer simulations were performed for two cases representing two different parameter approximations of the spring torque function. The first case uses piecewise linear functions to approximate $f(\theta)$. In the other case, a bi-linear form of spring torque function defines the parameter space whose components are the two slopes and the position of the nonlinear slope transition. Although the driving torque to the first rigid bar is the true input to the system, $\omega(t)$ can still be treated as the input to Equation (8.2). Since $\omega(t)$ is given and fixed, this study is then to investigate the relation between parameter space and response space $\mathcal{R}_X \subseteq \mathbf{R}^n$ constructed by X .

8.2.3.1 Case 1

Assume that the spring torque function $f(\theta)$ can be approximated by a piecewise linear function. Since this function is symmetric with respect to the origin,

it is represented by three line segments in the first quadrant. In Figure 8.3, the domain of $f(\theta)$ is discretized at $\pm\theta_1$, $\pm\theta_2$, and $\pm\theta_3$. In the experiment, the values of θ_1 , θ_2 , and θ_3 are 0.02, 0.03, and 0.05 (radians), respectively. Thus the new parameter space is a subset of \mathbf{R}^3 constructed by the vector $(f(\theta_1), f(\theta_2), f(\theta_3))$.

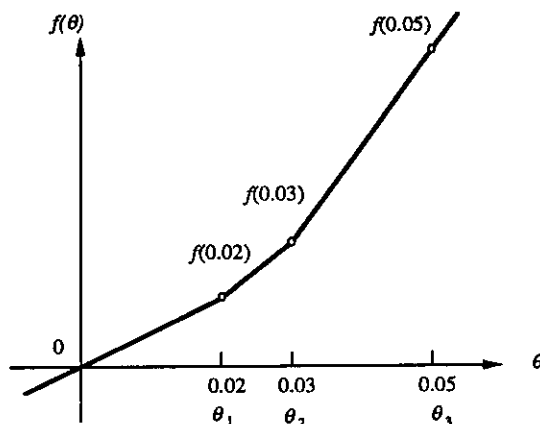


Figure 8.3: The piecewise linear function approximates the spring torque function.

Let us further assume the possible operational range of each parameter component to be that $f(\theta_1) \in [0.5, 0.6]$, $f(\theta_2) \in [0.75, 0.9]$, and $f(\theta_3) \in [1.25, 1.5]$ (lb-ft), as illustrated in Figure 8.4(a). In other words, the spring torque function of interest is bounded inside the shaded region between two slopes of 25 and 30. Then Figure 8.4(b) gives the new parameter space.

After applying Algorithm 4.1 to partition the parameter space, seven regions were obtained as shown in Figure 8.5, where the regions are labeled from 0 to 6. The threshold of partition variations was set equal to 0.00015 in terms of the variance of the squared Euclidean distances among system responses (see Appendix A for the definition of the scalar variance). A stratified sampling scheme was employed to sample 64 patterns for every pattern class. This partitioned result demonstrates that the system responses are more sensitive to the variation of $f(\theta_3)$ than to

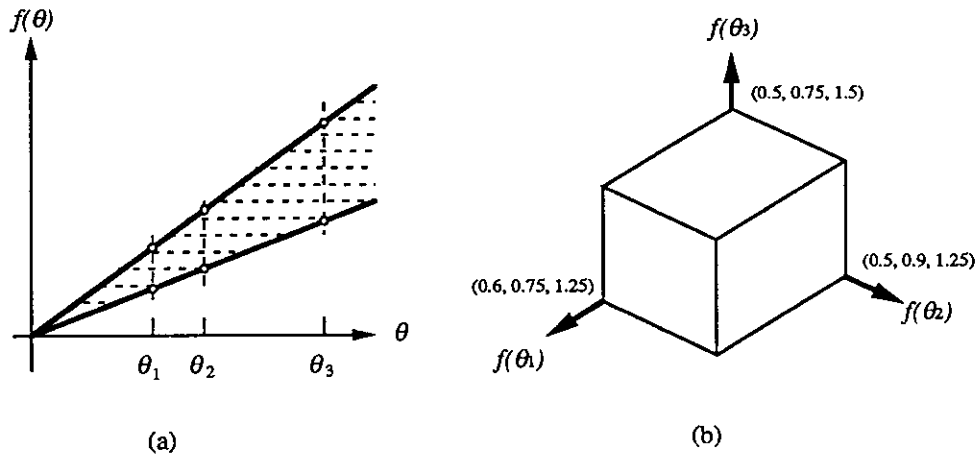


Figure 8.4: (a) The operational region of the spring torque function. (b) the new parameter space.

those of $f(\theta_1)$ or $f(\theta_2)$. Thus more partitions occur in the $f(\theta_3)$ dimension than other dimensions. The binary tree in Figure 8.6 presents the partition process, where the label under an internal node denotes the partitioned dimension of the parameter space. In Figure 8.7, the functional regions corresponding to classes of spring torque functions are presented.

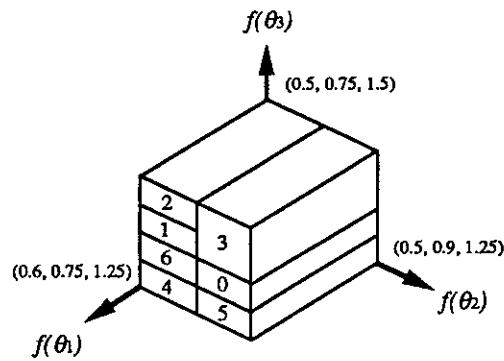


Figure 8.5: The partitioned parameter space, where the label on each cubic corresponds to its class identity.

Classification was carried out by a neural network based on the Hamming network model and trained by the MLVQ learning algorithm with three exemplars

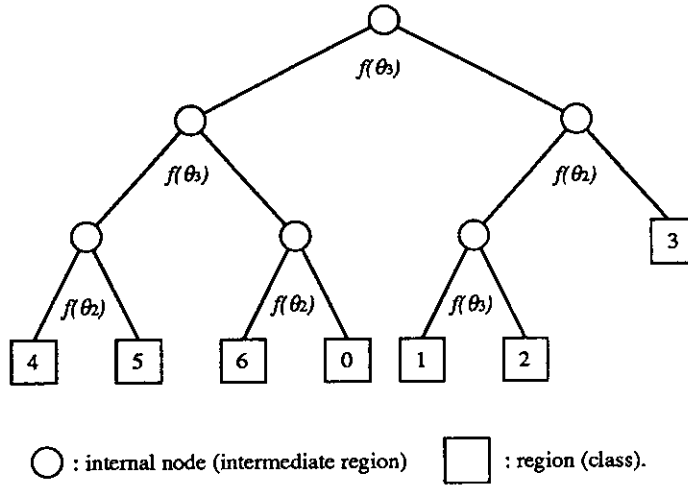


Figure 8.6: The binary tree of the partition process.

Table 8.2: Different levels of Gaussian noise and their corresponding SNR.

standard deviation	0.001	0.002	0.003	0.005
SNR	28.33	14.17	9.59	5.67
SNR (dB)	29.04	23.03	19.64	15.07

for each class. To examine the effect of noise on the classification performance, Gaussian noise with zero mean and various variances was added to the system responses of the patterns to be tested. This Gaussian noise is treated as measurement disturbances. Table 8.2 contains the approximate signal to noise ratio (SNR) corresponding to the Gaussian standard deviation which is set equal to the root mean squared (RMS) amplitude of the noise, where

$$\text{SNR} \triangleq \frac{\text{RMS of signal}}{\text{RMS of noise}}, \quad (8.4)$$

and

$$\text{SNR(dB)} \triangleq 20 \log_{10} (\text{SNR}). \quad (8.5)$$

Table 8.3 shows the percentage of misclassification for the training set in which

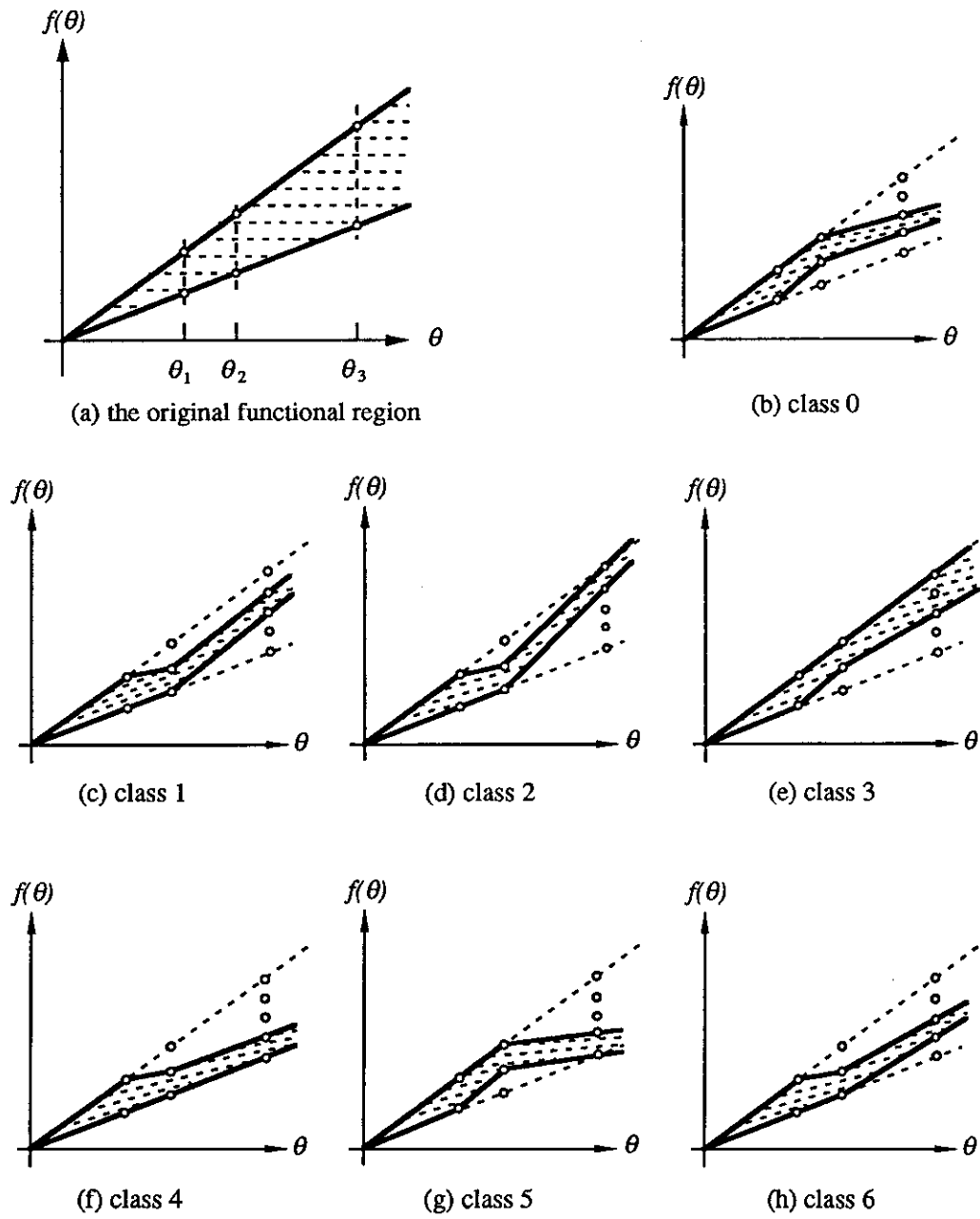


Figure 8.7: The classes of the spring torque functions correspond to the regions in the partitioned parameter space.

Table 8.3: Classification results in terms of error rates (%) for the training set and testing set contaminated by different levels of Gaussian noise.

Standard deviation of Gaussian Noise	0	0.001	0.002	0.003	0.005
Training set	0	6.39	12.04	18.66	29.4
Testing set	6	11	16.6	26.2	43.6

each class contains 64 training patterns. A testing set was also prepared in a different functional approximation form from that of the training set. This testing set contains 50 testing patterns of bi-linear functional approximation as depicted in Figure 8.8. The constraints of these testing patterns are that the second slope is greater than the first slope ($\alpha < \beta$), and the slope transition occurs within a certain interval ($\theta_0 \in [0.02, 0.035]$). It is worth noting that only 0.1% of the testing patterns was classified with class 0 and class 5 in which $\alpha > \beta$, even though these patterns were corrupted by Gaussian noise with $\text{SNR} = 14.17$. The classification result provides very useful information for constructing a better model structure of $f(\theta)$, because the case of $\alpha > \beta$ can be eliminated and the whole parameter space is further reduced.

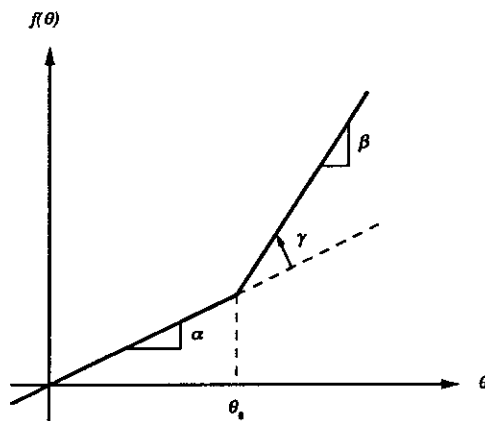


Figure 8.8: A bi-linear approximation of the spring torque function.

The reasons that the testing set is more sensitive to noise can be explained as:

1. The model structure used in the testing set is different from the one used in the training set.
2. The patterns generated for the testing set are not uniformly distributed in the parameter space of the training patterns.

8.2.3.2 Case 2

The spring torque function is assumed to be bi-linear as depicted in Figure 8.8. Thus the new parameter space is defined to be a subset of \mathbf{R}^3 with components $(\theta_0, \alpha, \gamma)$, where $\gamma = \beta - \alpha$ and $\alpha < \beta$. The ranges of these parameters are defined as follows: $\theta_0 \in [0.02, 0.035]$, $\alpha \in [25, 27]$, and $\gamma \in [0, 4]$. Applying Algorithm 4.1, Figure 8.9 shows the partitioned parameter space with 6 regions, labeled from 0 to 5. The variation threshold for partitioning was set equal to 0.00017 in terms of the maximal Euclidean distance of sampled patterns to the mean pattern vector of the region. The systematic sampling scheme was used to sample patterns inside the parameter space, with 64 patterns for each class.

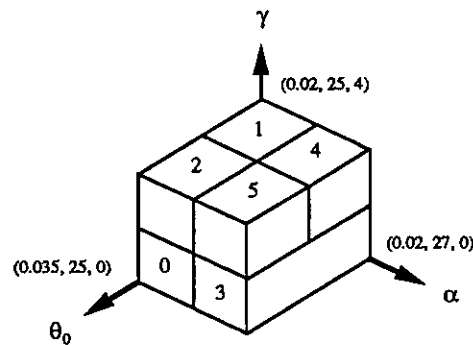


Figure 8.9: The partitioned parameter space.

This partitioned result demonstrates that the system responses are less sensitive to the variation of θ_0 when $\gamma \in [0, 2]$. In other words, when the difference between

Table 8.4: Classification results in terms of error rates (%) for the training set and testing set corrupted by different levels of Gaussian noise.

Standard deviation of Gaussian noise	0	0.001	0.002	0.003	0.005
Training set	0	20.21	36.9	45.7	56.96
Testing set	8	21.8	35	43.8	54.6

the two slopes, α and β , is large (or $\gamma > 2$), the effect of the variation of θ_0 on the system responses becomes important. It is to be expected that the position of nonlinear slope transition gets more crucial for a larger γ . Therefore a partition occurs in the θ_0 dimension when $\gamma \in [2, 4]$. The partitioned result thus shows the relative sensitivity of the system responses to the parameters. Sensitive portions in the parameter space are eventually divided into smaller class regions, such that the patterns in one class vary within a certain threshold. Figure 8.9 presents a reasonable sensitivity result via the top down dichotomy algorithm.

Table 8.4 presents the error rates of the classification for the training set and testing set in the presence of Gaussian noise. A testing set with 50 patterns was generated randomly in the same parameter space. In this case these testing patterns have the same functional form as the training set. Both sets give very close estimations of classification results. Hence the training set provides a plausible estimation of the classification performance provided that the training patterns are sampled uniformly from the parameter space. Although perfect classification was obtained for the training set under noise free condition, noise greatly affects the classification performance. The possible reasons for the poor results are listed:

1. Although the parameter space is divided into disjoint regions, the system response space corresponding to different classes may contain overlap regions. These overlaps degenerate the classification performance.

2. The linear feature extractor is not effective enough.
3. The pattern classifier did not perform well.
4. Pattern classes are partitioned by hyperplanes as boundaries in the parameter space. However, the nature structure of system responses does not possess such precise and regular boundaries. The decision regions in the system response space are highly irregular.

Since the classifier training was effective to obtain perfect classification on the training set without noise, the first three reasons are ruled out. In comparison to Case I, item 4 listed above seems to be the main reason of poor classification performance. Hence, it is important to define an effective parameter space. It turns out that the approximation by piecewise linear functions is an effective approach to defining the parameter space.

The results presented in Case I are very plausible and demonstrate the applicability of the methodology proposed in this study.

8.3 The Need for the Pattern Recognition Approach

In this section, we investigate the situations for which the optimization approach fails, while the PR approach provides useful information. The Downhill Simplex Method (DSM) [PFTV88] was implemented as an optimization procedure to estimate the parameters of the simple articulated joint. The DSM requires only function evaluations, not derivatives. Equation (8.2) does not need to be transformed into a difference equation. The function evaluation of the DSM is performed by calculating the mean square error between the true system response and the simulated system response of the model.

The basic idea of the DSM to search a minimum is

1. construct a polyhedron with $n + 1$ vertices in an n -dimensional vector space,
2. repeatedly flip the highest point downhill until a local minimum is reached,
3. shrink the size of the polyhedron as it descends into the basin of a valley.

8.3.1 Experiment Design

The true spring torque function is assumed to be a bi-linear function as shown in Figure 8.8 with $\alpha = 26.9$, $\beta = 38.1$, and $\theta_0 = 0.029$. Notice that this function is symmetric to the origin and the working area of this case is in the third quadrant.

The model structure of the spring torque function for the DSM was selected to be a polynomial

$$f(\theta) = a\theta + b\theta^2 + c\theta^3 \quad \text{for } \theta \leq 0 \quad (8.6)$$

where a , b , and c are the parameters to be identified by the DSM.

8.3.2 Simulation Results

The true system response is classified with a functional region by the PR approach shown in Figure 8.10 where the region is in between two dotted lines and the true spring torque function is depicted in a solid line. This Figure only depicts the spring torque function in the first quadrant. Even when noise is presented with $\text{SNR} = 5.67$, the classification result is still correct.

Figure 8.11 presents the identification result of the spring torque function by the DSM. In fact, the identified functions varies slightly with or without noise added to the true system response. Obviously, this function does not fall inside the functional region obtained via the PR approach. It lies inside another functional region where its simulated system response is indeed classified by the pattern classifier, see Figure 8.12. It is easy to see that the best model obtained via the

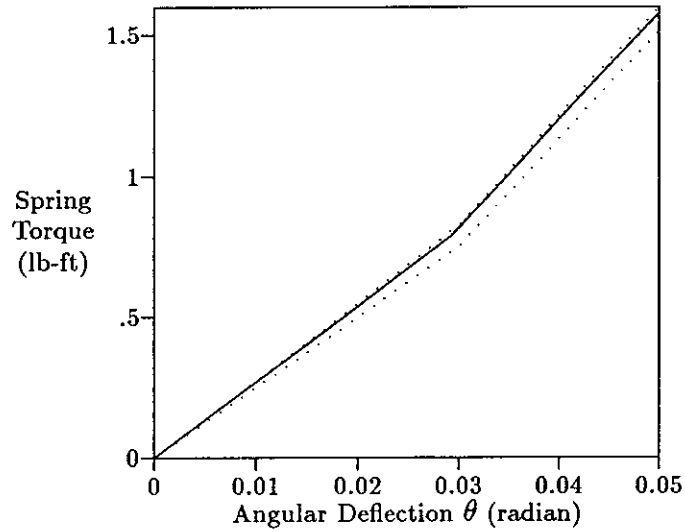


Figure 8.10: The true spring torque function (solid line) and the functional region identified via the PR approach

optimization search approach using a third order polynomial approximation is not acceptable.

8.3.3 Discussion

Actually, the true spring torque function for this case cannot be accurately approximated by a low order polynomial. Thus, the DSM starts with an inappropriate model structure. On the other hand, a bi-linear function with $\theta_0 = 0.30$ may be selected as a starting model structure for the DSM after a functional region is obtained via the PR approach. Even though θ_0 is different to the one for the true function, the identified result is still better than the result obtained from a polynomial model structure in terms of mean square errors between the true system response and the simulated system response of the identified models. This demonstrates that the PR approach can provide more information in selecting model structure.

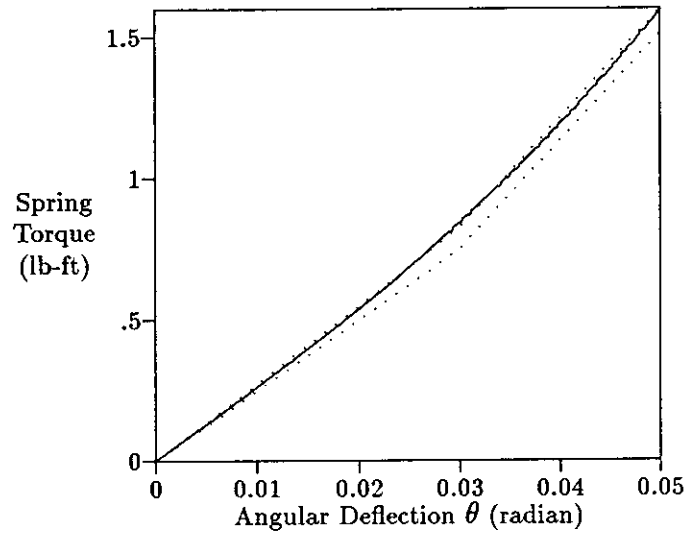


Figure 8.11: The spring torque function obtained via the DSM (solid line) and the functional region identified via the PR approach

In addition, the functional region can be used to cross-validate the spring torque function obtained via the optimization approach, especially when noise is presented. The simulated system response of the identified model can also be used as an input to the PR approach. The classification result is then compared with the result of the observation from the true system.

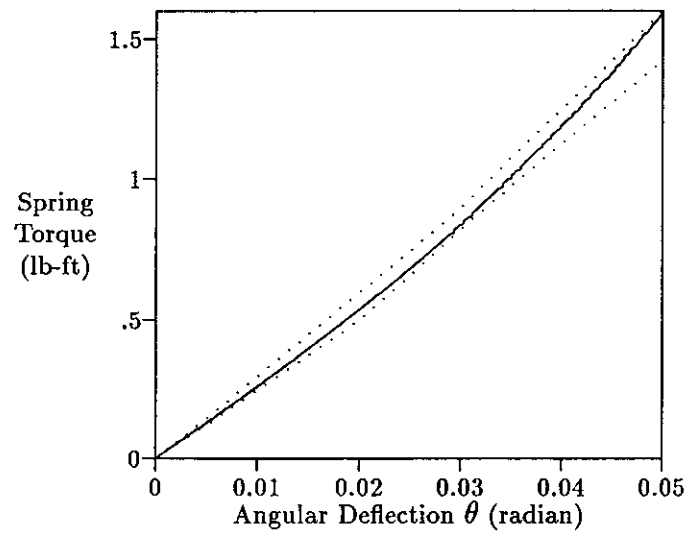


Figure 8.12: The spring torque function obtained via the DSM (solid line) and the functional region with which it is classified via the PR approach

CHAPTER 9

Space Station Freedom Model

9.1 Introduction

On-orbit parameter identification is indispensable for the development and verification of large, flexible multibody, controlled spacecraft because such spacecraft are unstable in 1-g environment and cannot be satisfactorily tested before flight.

One of the most difficult tasks in the identification of nonlinear structural parameters is to determine the most probable mathematical descriptors of the nonlinearities in question. A nonlinear structural parameter may assume any of a large number of candidate descriptors from which one must choose the most likely candidate. The methodology proposed in the dissertation is applied below to the realization of nonlinear structural parameters of the Phase I Space Station Freedom (SSF) model.

9.2 The Space Station Freedom Model

9.2.1 The System

Figure 9.1 presents the SSF model consisting of a central body and a starboard and port bodies, all modeled as flexible bodies. Solar panels are attached to the extraneous bodies which are connected to the central body by one degree-of-freedom Alpha gimbals. The simulated maneuver depicted a transient rotation of the solar arrays to achieve perpendicularity with respect to the sun line, while maintaining the central body in a predetermined attitude control mode. The Alpha

gimbals are thus critical components to the system. The identification of a model for them is essential to the design of the gimbal control system.

Our on-orbit experimental data were obtained analytically by simulation. The space station simulator was written at TRW in the FORTRAN 77 programming language and ported to the Sun 4 computers of UCLA Computer Science Department.

9.2.2 Objective

The objective of this effort is to realize the mathematical characteristics of the disturbance damping torques at the Alpha gimbals as functions of the gimbals' angular velocities.

9.2.3 Experiment Design

In the selected maneuver, the extraneous bodies are rotated 15 degrees relative to the central body, whereupon the gimbal control system is suddenly shut off. The free vibrations that ensue are monitored by angular accelerometers mounted to the solar panels and sampled every 0.01 second for a period of 50 seconds. Figure 9.2 shows a typical system response in the time domain for the aforementioned Space Station maneuver.

9.2.4 Experiment Data

The application of our methodology, to the realization of the disturbance damping torque at the Alpha gimbals, focused on the time domain response because of its sensitivity to variations of this parameter. By contrast, we found that the damping torque does not affect the frequency of the system response significantly and, therefore, frequency domain analysis techniques are not particularly useful in

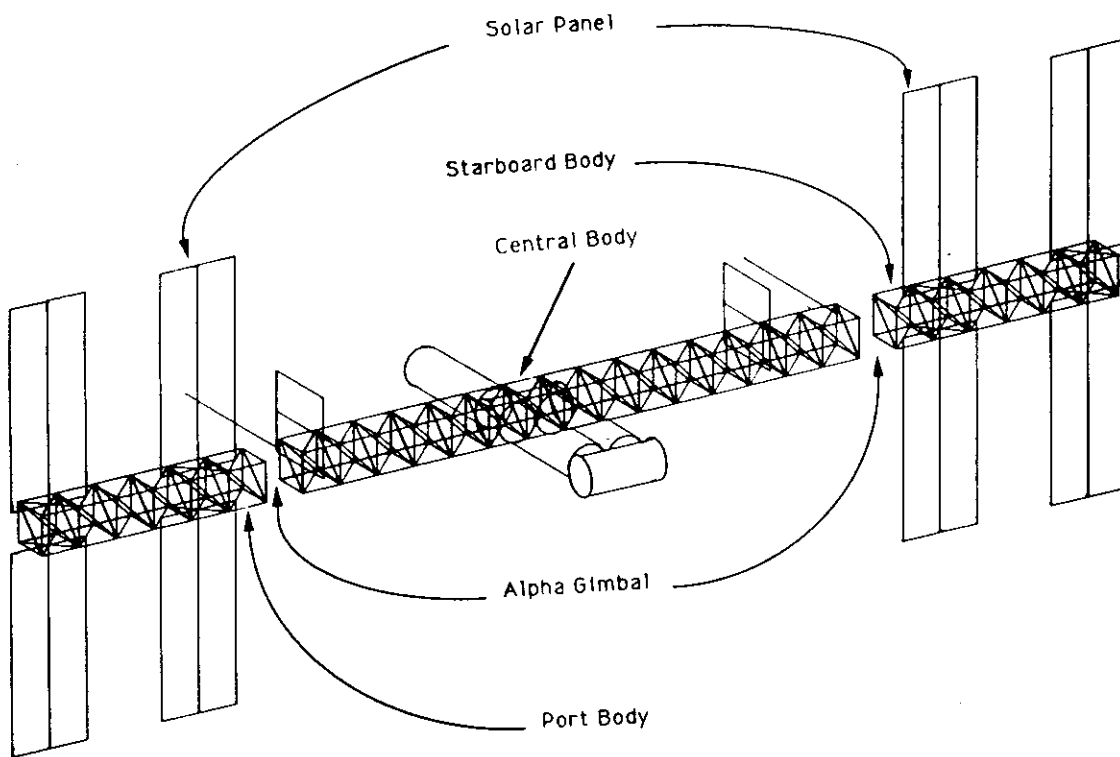


Figure 9.1: The Space Station Freedom model.

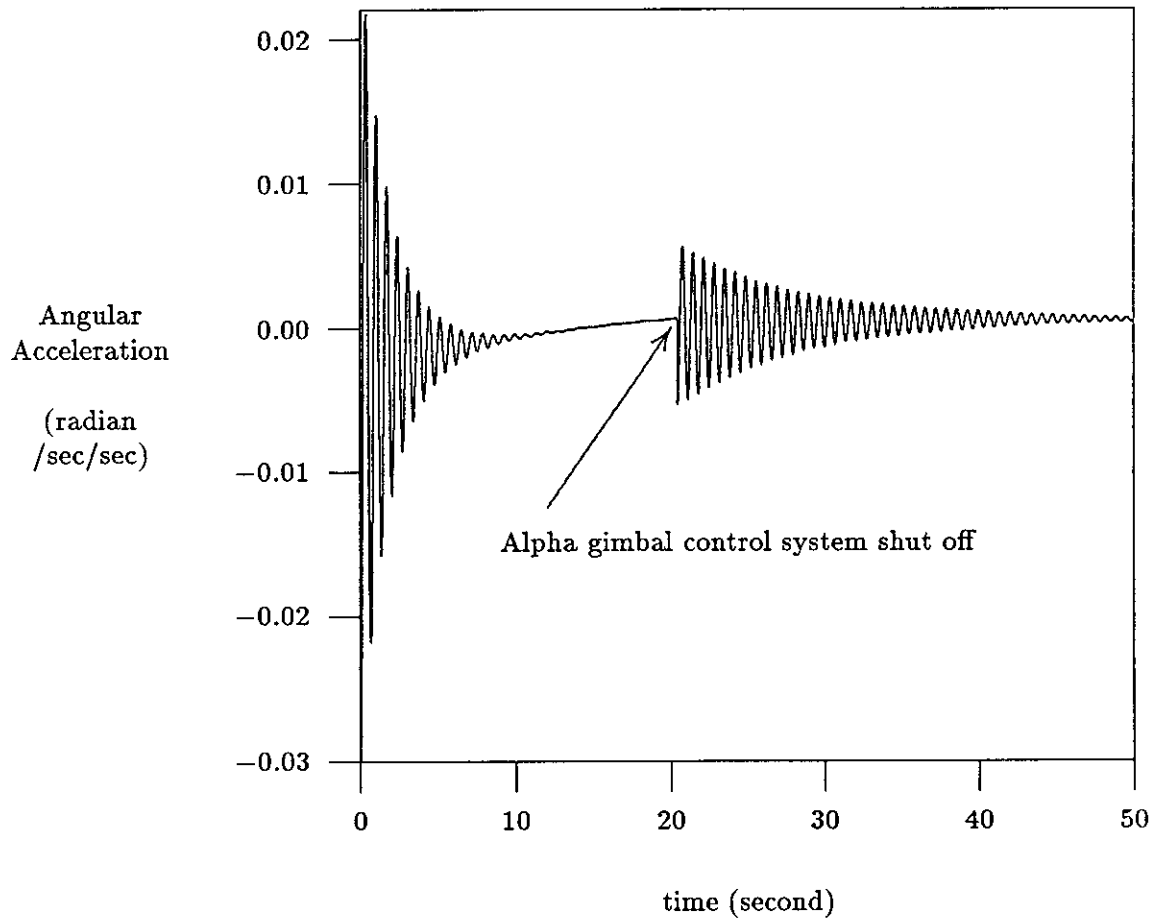


Figure 9.2: A system acceleration response in the time domain.

facilitating classification.

Our methodology utilizes the logarithm of the power density of the acceleration decay signal averaged over time. The predominant flexible mode in the system response has a frequency of 1.47 Hz (see Figure 9.2). The basic element of the processed data is the squared sum of the system response over a time frame window of 1.36 seconds, which constitutes a two-cycle period of the predominant mode. The decay signal is given, then, as:

$$q(t) = \sum_{i=0}^{135} a(t + i\Delta t) * a(t + i\Delta t), \quad (9.1)$$

where $\Delta t = 0.01$ second and $a(t)$ is angular acceleration of the system response measured at time t . A moving average technique is applied to smooth the decay signal, yielding:

$$s(t) = \frac{1}{M + N + 1} \sum_{i=-M}^N r(t + i\Delta t), \text{ for } t \geq T + 0.25 \text{ second}, \quad (9.2)$$

where $r(t) = \log(q(t))$, $M = N = 25$ are empirical constants, and T is the time when the gimbal control system shuts off. The signal used in this application of our methodology was sampled from $s(t)$ at 256 points.

9.2.5 Classification Results

The disturbance damping torque of interest was assumed to be confined to the shaded region shown in Figure 9.3. Figure 9.4 illustrates the classes of functional regions obtained via the top down dichotomy algorithm.

The classification results were generated in a noisy environment, where various Gaussian noise levels were added to the system response. Two types of noise were examined. One was a Gaussian noise with a fixed standard deviation. The other was a Gaussian noise with a standard deviation proportional to the amplitude of the signal.

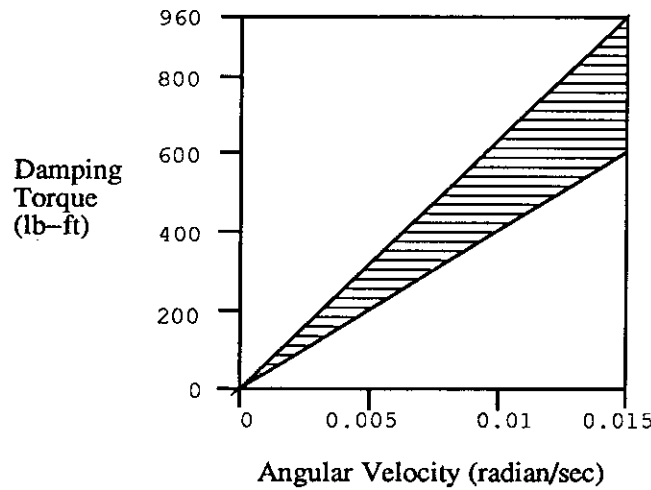


Figure 9.3: The possible region of the unknown damping torque function.

Table 9.1: Classification results with noise of fixed standard deviations.

standard deviation	0	0.00005	0.0001	0.00015	0.0002
SNR [†]	∞	30	15	10	7.5
percent of correct classification	100	100	92.19	79.69	62.75

$$\dagger \text{SNR} \triangleq \frac{\text{RMS of signal}}{\text{RMS of noise}}$$

9.2.5.1 Gaussian Noise with A Fixed Standard Deviation

Table 9.1 presents percentages of correct classification obtained in the presence of Gaussian noise characterized by fixed standard deviations. The overall signal to noise ratio (SNR) in the table was calculated from the standard deviation value which is set equal to the root mean squared (RMS) amplitude of the noise. It should be noticed that over 90 percent of correct classification was obtained in the presence of noise with an overall SNR of 15. Figure 9.5 presents the temporal responses of the SSF model with and without noise.

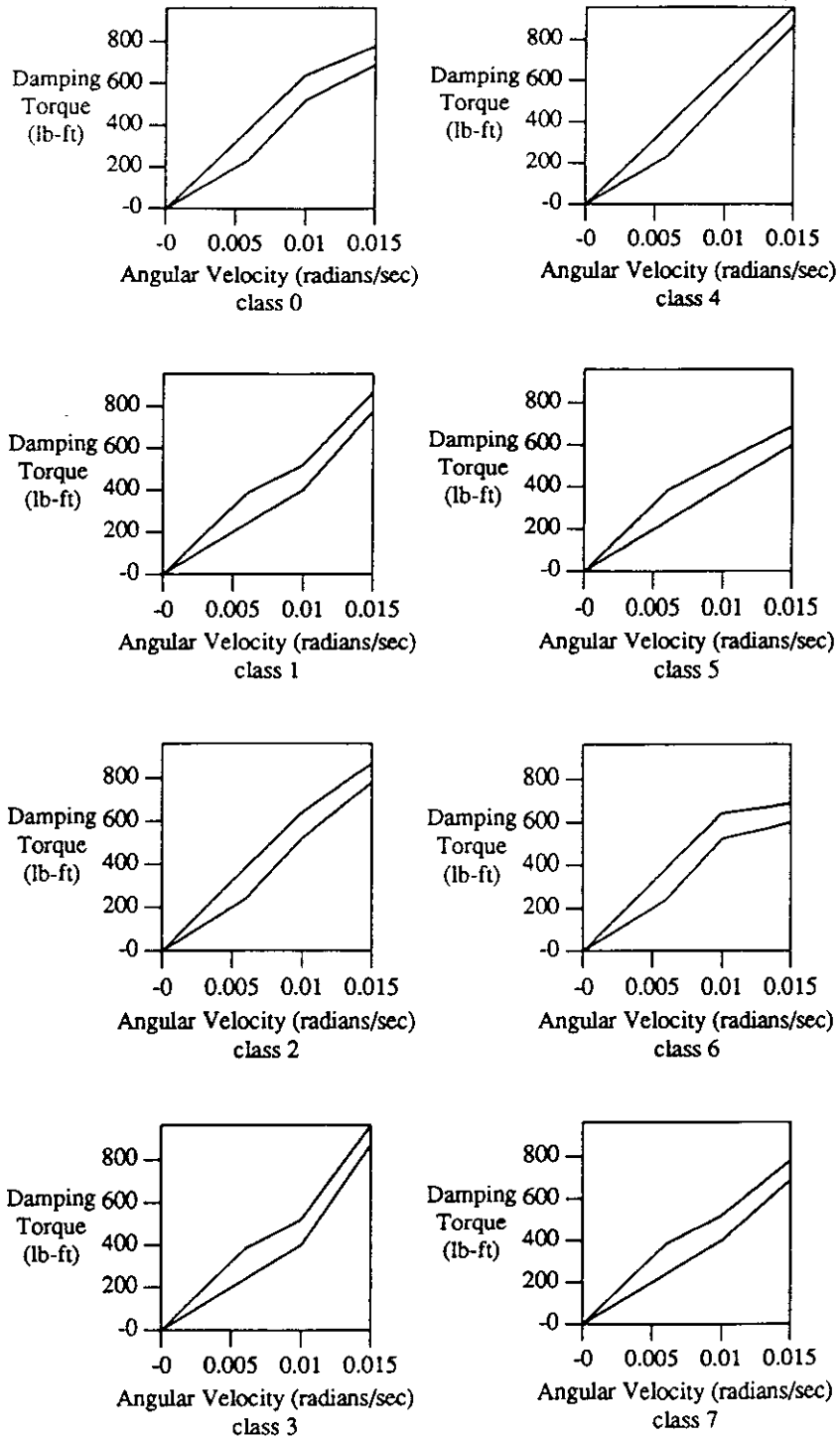
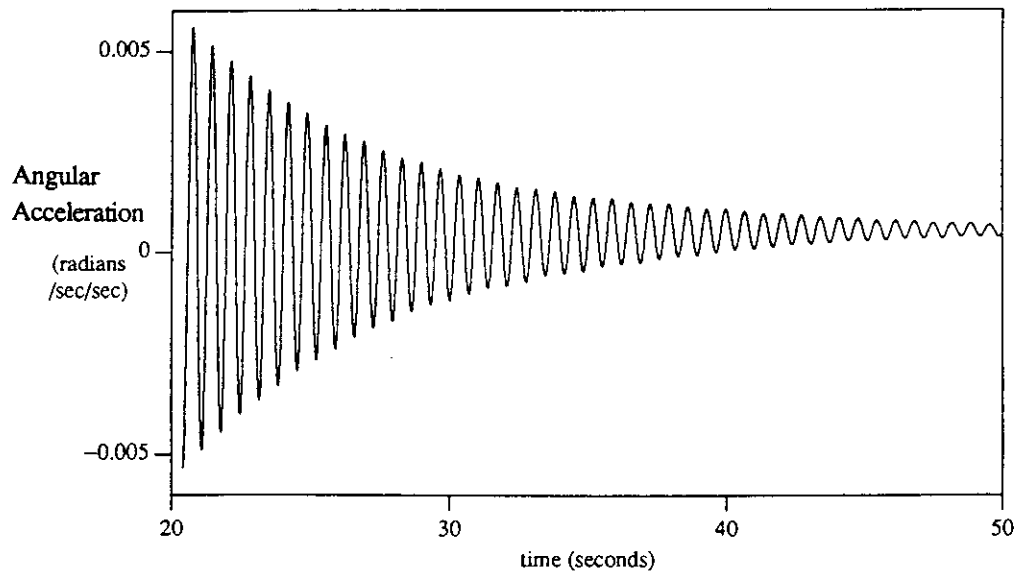
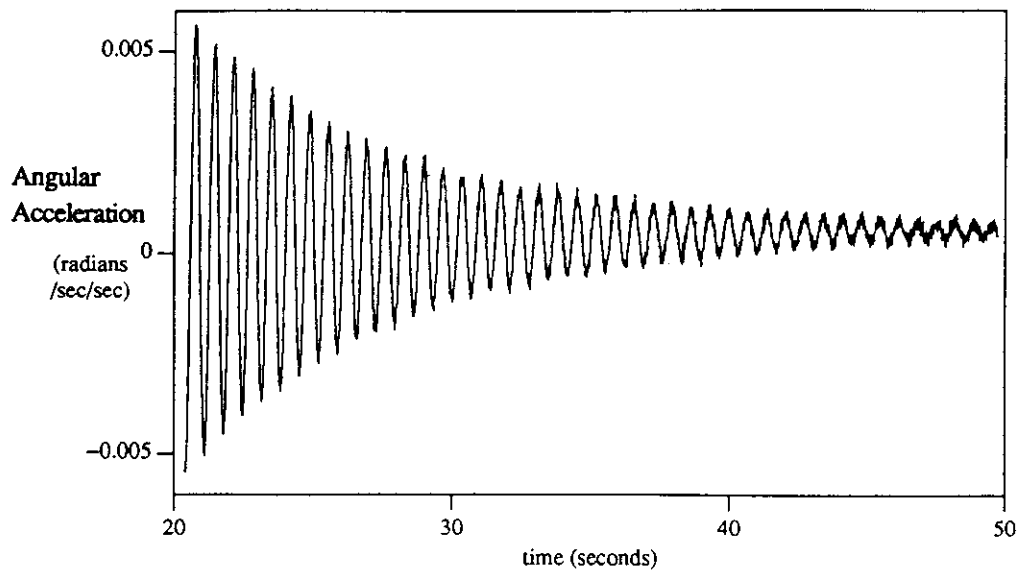


Figure 9.4: The pattern classes.



(a) A system response without noise.



(b) A system response with Gaussian noise of standard deviation 0.0001.

Figure 9.5: A system response with and without noise.

Table 9.2: Classification results with noise proportional to signal's amplitude

noise level (%) of signal	5	10	15	20
percent of correct classification with MNL = 0.00005	100	95.31	89.06	85.94
percent of correct classification with MNL = 0.0001	90.62	90.62	87.50	79.69

9.2.5.2 Gaussian Noise Proportional to the Signal's Amplitude

Table 9.2 contains classification results obtained in the presence of Gaussian noise where the RMS amplitude of the noise is proportional to the signal's amplitude. As the signal decays (see Figure 9.2) the noise RMS amplitude is reduced until a minimum threshold, defined as minimum noise level (MNL), is reached; thereafter, the noise is maintained at the MNL, regardless of the signal's amplitude. As shown in Table 9.2, our realization methodology yields high percentages of correct classification in the presence of noise levels as high as 20 percent of the signal's amplitude. Figure 9.6 shows the temporal system response with noise levels of 10 and 20 percent of the signal's amplitude and MNL standard deviation = 0.0001.

9.3 Conclusion

The proposed methodology was successfully applied to the characterization of the damping torques at the Alpha gimbals of a realistic model of the SSF. The results show a high percentage of correct classification in a noisy environment. Two types of Gaussian noise, with fixed standard deviation and standard deviation proportional to the signal's amplitude, were added to the artificial data for

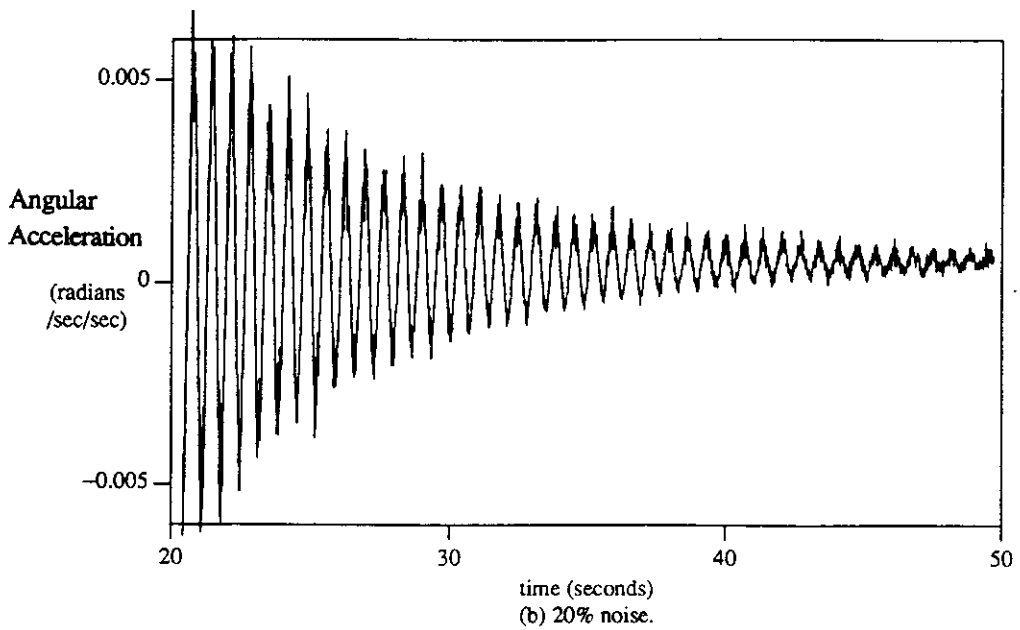
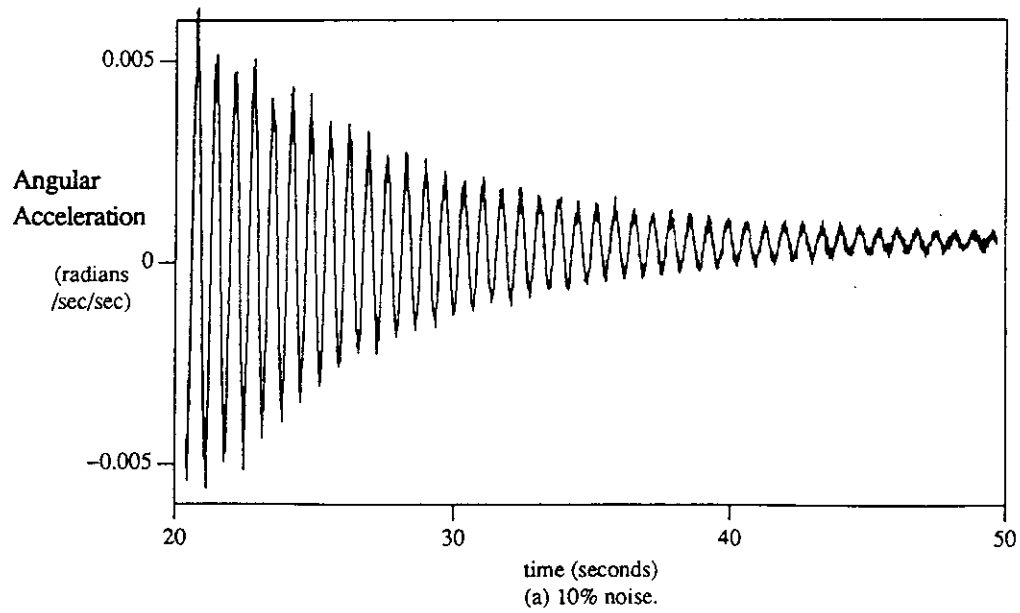


Figure 9.6: A system response with 10% and 20% noise at $MNL = 0.0001$.

examination. About 80% of correct classification was obtained in the presence of 20% Gaussian noise or noise with an over all SNR of 10. In fact, the typical SNR in many engineering applications is around 15. The results are very encouraging, showing a high level of confidence on classifications.

In the past two years, TRW Space & Technology Group has concentrated its efforts in the realm of structural parameter identification on the Dynamic Programming Filter (DPF), a member of a class of nonlinear filtering techniques based on modern control theory [Sim88]. One of the most difficult tasks in the identification of nonlinear structural parameters is to determine the most probable mathematical descriptors of the nonlinearities in question. In reality, a nonlinear structural parameter may assume any of a large number of candidate descriptors from which one must choose the most likely candidate. The identified results of the proposed methodology, which characterizes the nonlinear structural parameters, are very useful to the DPF. As shown in Figure 9.4, the identified functional region confines the true damping torque function of the Alpha gimbal and characterizes its nonlinearity with a very high reliability.

The PR system has explored a plausible functional region for the nonlinear damping torque. The neural network classifier has learned and memorized the typical system responses of different model classes. Any change of the characteristic of the Alpha gimbals can be detected rapidly. This unique property of the methodology presented in this dissertation is very important to cope with the critical system components whose functional characteristics may change due to the heating and aging processes.

CHAPTER 10

Aircraft System

10.1 Introduction

System identification plays an important role in the design and analysis of aircraft stability and control. Identification results provide information for analyzing aircraft stability, handling qualities, and control systems. Nowadays, high-fidelity flight simulators are increasingly necessary in modern flight research and pilot training. These simulators require accurate models based on adequate identification results as well as complete stability and control data.

This chapter examines the application of the proposed parameter characterization technique to the problem of characterizing aircraft stability and control derivatives from flight test data.

10.2 The Aircraft System

10.2.1 Aircraft Reference Coordinate Systems

Several coordinate systems are used in aerospace applications, including the body-axis, vehicle-carried vertical, and the air flow systems.

Figure 10.1 shows the body-axis system whose origin is at the center of gravity. The positive X axis points forward, the positive Y axis points to the right, and the Z axis points down. The body axes are fixed at the center of gravity and, hence, rotate and move with the aircraft. The body-axis angular rates are defined as the projections of the angular velocity on the body axes. The roll rate p , pitch rate q ,

and yaw rate r are the angular velocities in the body X , Y , and Z directions. The right-hand rule defines the sign conventions, illustrated in Figure 10.1.

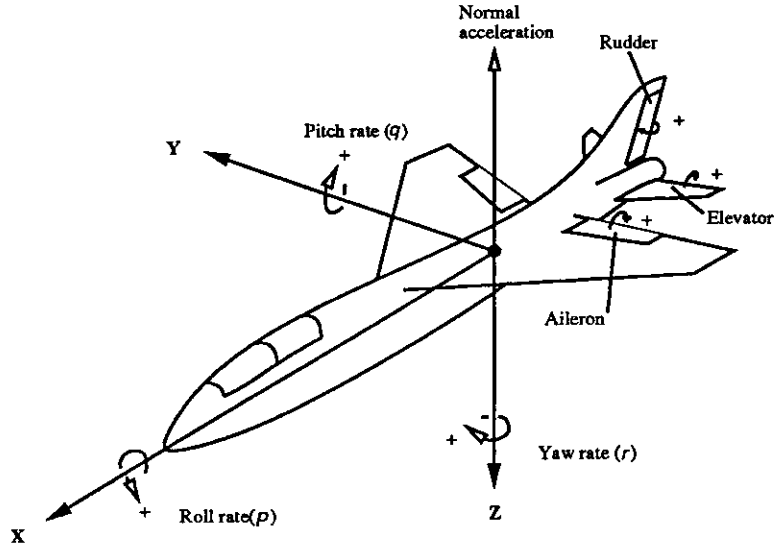


Figure 10.1: The aircraft body-axis system and control surfaces.

The vehicle-carried vertical axis system also has its origin at the center of gravity of the aircraft, depicted in Figure 10.2. The positive X_V axis points north,

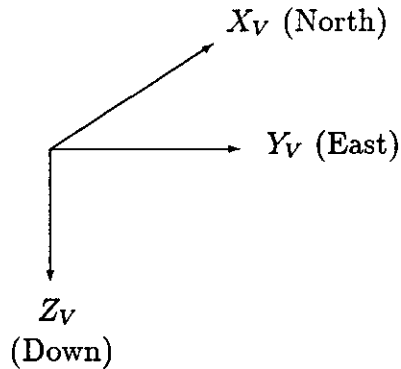


Figure 10.2: Vehicle-carried vertical axis system.

the positive Y_V points east, and the positive Z_V points down. The attitude of the aircraft with respect to the earth is defined by three Euler angles ψ (heading

angle), θ (pitch attitude), and ϕ (roll attitude) in terms of the orientation of the aircraft body axes with respect to the vehicle-carried vertical axes. Figure 10.3 shows the rotations required to transform the vehicle-carried vertical axes to the body-axes. The order of rotation, listed from Figures 10.3(a) to (c), is important. The positive ψ is a clockwise rotation about the Z_V axis to define a new axis system (X_1, Y_1, Z_1) , where Z_1 is identical to Z_V . The pitch attitude θ is a rotation about the Y_1 axis into the axis system (X_2, Y_2, Z_2) . Finally, the roll attitude ϕ is a rotation about the X_2 axis to arrive at the body axes (X, Y, Z) .

The velocity of the aircraft relative to the air flow is defined to be the wind-relative velocity whose vector components in the body-axes X, Y, Z are u, v , and w , respectively. Hence the total wind-relative velocity is

$$\|V\| = \sqrt{u^2 + v^2 + w^2}. \quad (10.1)$$

The angle of attack α and the angle of sideslip β are defined by

$$\alpha = \tan^{-1} \frac{w}{u}, \quad (10.2)$$

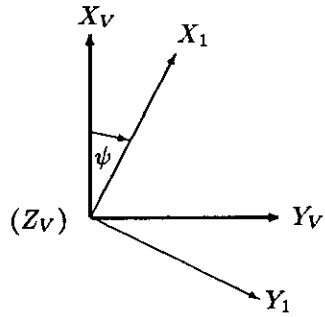
and

$$\beta = \sin^{-1} \frac{v}{\|V\|}. \quad (10.3)$$

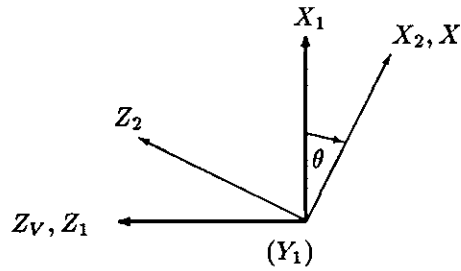
Figure 10.4 shows the flow angles and the relationship between the body and the wind axes.

10.2.2 Control Surfaces

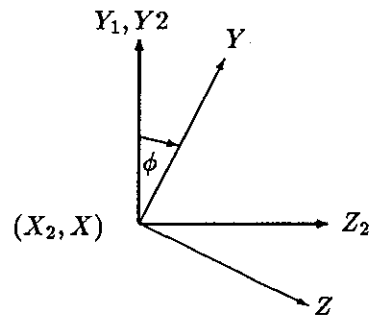
The positions of the control surface are the inputs to the aircraft stability and control equations. The ailerons, elevator, and rudder are usually the basic aircraft control surfaces as shown in Figure 10.1. All the surface positions are expressed as the angular deflections measured normal to the hinge line. Positive deflection



(a) Rotation through ψ about Z_v axis which points into the page.



(b) Rotation through θ about Y_1 axis.



(c) Rotation through ϕ about X_2, X axis.

Figure 10.3: Rotation of axes through Euler angles.

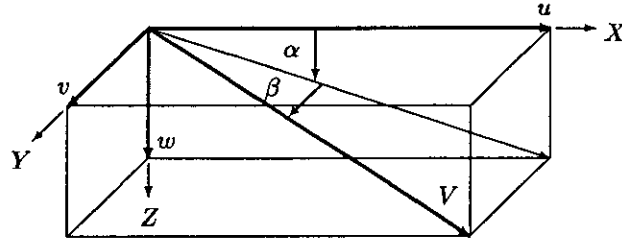


Figure 10.4: Flow angles and relationship of body and wind axes.

is trailing edge down or trailing edge left. Aileron deflection is defined to be the left surface position minus the right surface position.

10.2.3 Equations of Motion

The derivation of the aircraft rigid-body equations of motion begins with Newtonian mechanics. The basic equations to the rotating body-axis system are the momentum and angular momentum equations:

$$F = \frac{d}{dt}(mV) + \omega \times (mV) \quad (10.4)$$

$$M = \frac{d}{dt}(h) + \omega \times h \quad (10.5)$$

where F is the external applied force, M the external applied moment about the center of gravity, V the velocity vector, ω the angular velocity vector of the body-axis system with respect to the inertial space, and h the angular momentum vector about the center of gravity. By definition, the components of ω are (p, q, r) in the body-axis system.

The external applied forces include aerodynamic forces, gravity, and engine thrust. The external applied moments consist mainly of aerodynamic moments (including rolling, pitching, and yawing moments) and gyroscopic moment from

the rotating machinery in the engine. We can derive the aerodynamic forces and moments in terms of the dynamic pressure, the aircraft dimensions, and the nondimensional coefficients. These coefficients are functions of the aircraft states and controls. In general, they are nonlinear. The parameters of the parameterized coefficients are called the stability and control derivatives. For example, the locally linearized approximation of the lift coefficient C_L , described below, can be expressed as:

$$C_L = C_{L_\alpha}\alpha + C_{L_\delta}\delta + C_{L_b}, \quad (10.6)$$

where δ is a generic notation of controls and C_{L_α} , C_{L_δ} , and C_{L_b} are the stability and control derivatives. Substituting the external applied forces and moments into Equations (10.4) and (10.5), we can obtain the aircraft equations of motion. The detail derivation of equations of motion can be found in most aircraft dynamics texts, for example, [Etk82, DAK88].

The lift is the component of the combined aerodynamic forces on an aircraft perpendicular to the aircraft velocity vector and is directed upward. To most aircraft, the lift sustains the weight of the aircraft so that the aircraft can fly. The lift coefficient C_L defines the proportion of the lift to the dynamic pressure and some aircraft reference area. Therefore it is an important coefficient to the aircraft stability and control. One application of this study is to characterize this coefficient as a function of the angle of attack α .

10.3 F-15 Flight Simulator

The F-15 flight simulator was written in the FORTRAN programming language by Eugene L. Duke et al. of Dryden Flight Research Facility, Edwards, California in 1977. This simulator was based on a linear aircraft model described in [DAK88]. In 1990, Bob Tisdale installed part of the simulator on the Sun computer of Computer

Science Department of UCLA. Some subroutines that control I/O interfaces and govern lateral directional equations of motion were discarded in order to reduce its size.

In Figure 10.5, the flow chart illustrates the major steps of the flight simulator. The left part of this chart consists of the procedures performing the initialization of the simulator. The right part conducts the simulation task. The simulator begins with initialization of system state variables. It then inputs the aerodynamic data which are, in fact, look-up tables as the implementation of the aircraft stability and control derivatives. For example, the lift coefficient is implemented as a three dimensional table with three components of aircraft velocity in terms of Mach number, elevator deflection, and angle of attack to index a table entry. The table look-up task is carried out by a linear interpolation among adjacent table entries. These aerodynamic data are different from one aircraft to another. Thus, the tables were modified in order to generate the training set in this study. Step 3 initialize the control system variables.

The steps encapsulated by a dashed box from step 5 to step 10 is the procedure MODELS which calculates the control system, equations of motion, observation variables, and so on. The procedure MODELS is called in both the initialization and simulation stages. Step 4 executes 100 times of MODELS. The number of iterations, 100 here, is not so crucial and only means “many.” Since the control system performs feedback control consisting of many first order and second order variables, the purpose of 100 iterations is to stabilize these control variables. Step 5 calculates air data parameters such as Mach number which depends on the height above the sea level. Step 6 computes feedback control variables, actuator outputs, and control surface positions. Step 7 computes force and moment coefficients, like the lift coefficient C_L , which require table look-up operations and linear

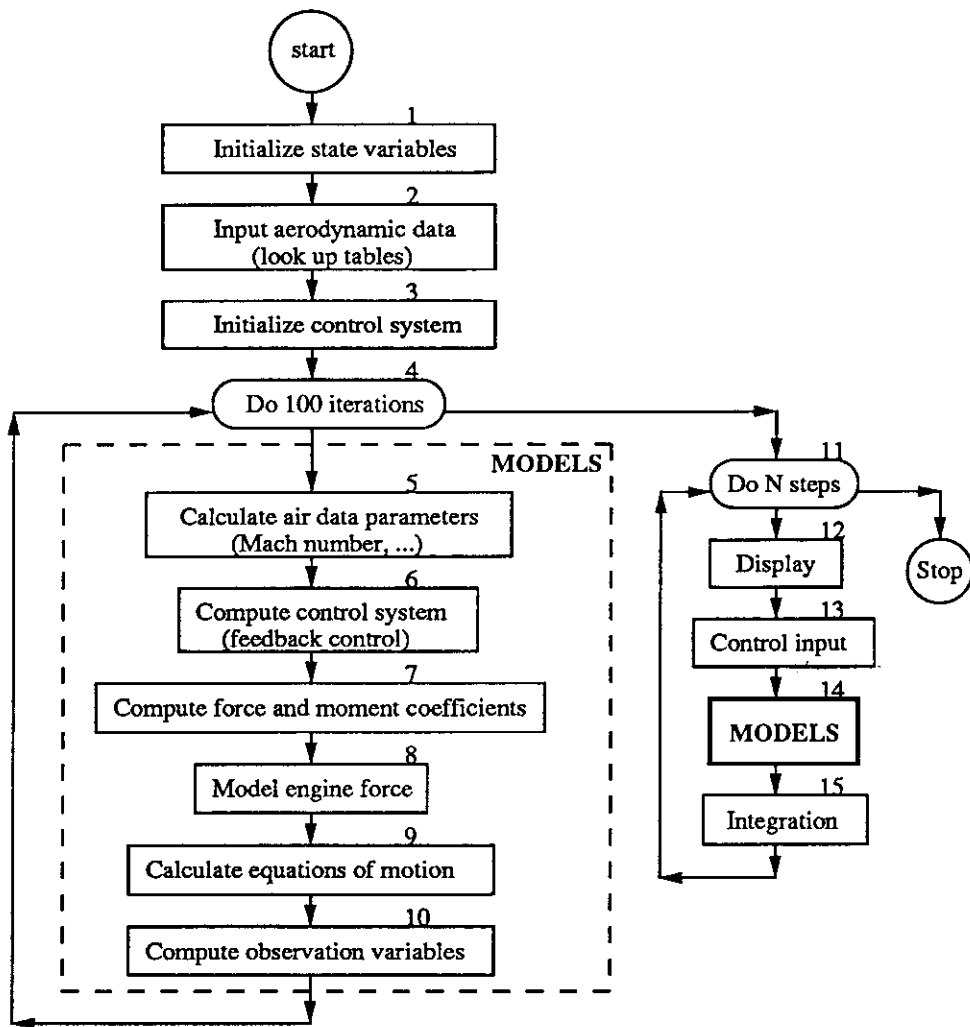


Figure 10.5: The flow chart of the F-15 flight simulator.

interpolations. Step 8 calculates the effects of engines such as thrusts, torques, and gyroscopic effects.

Step 11 performs the flight simulation, where the number of step N is specified by the user. At each simulation time step, some observation variables are displayed first. Then step 13 inputs the control parameters such as pilot stick positions and engine thrust. It is followed by the procedure MODELS. Finally, step 15 integrates the differential equations using modified Euler method (or second order Runge-Kutta method).

The training data corresponding to the responses of aircraft with different lift coefficients were generated by the flight simulator with modified look-up tables for the lift coefficient. Before utilizing the simulator, it is worth noticing that the inputs to the aircraft stability and control equations are the control surface positions while the inputs to the simulator are the pilot maneuvers. Because of the feedback control system, the control surface positions depend on the current status of the aircraft, control system variables, and pilot maneuver. Hence, give the same pilot maneuver, aircraft with different coefficients (actually different aircraft) generate different responses. It turns out that these different responses result in different control surface positions via the feedback control system. Even with the same pilot maneuver, the inputs to the system equations with different coefficients that are in fact different models are different. In order to characterize the coefficients of the aircraft model, it is required that the inputs or controls to the different models should be identical. Thus the inputs to the flight simulator for generating training data were modified to be the control surface positions; and the feedback control system was bypassed.

10.4 The Objective

The objective of this application is to characterize the lift coefficient C_L of the F-15 as a function of the angle of attack α . In this study, only the longitudinal motion is considered.

10.5 Experiment Design

The primary condition for experiment design is the safety of the aircraft. To estimate stability and control derivatives is one of the major purposes of flight test which may justify the design of flight test maneuvers. However, the design of a flight test maneuver involves many complex factors including hardware limits and pilot involvement. The maneuvers should be designed such that the pilot can fly with the required precision. In addition, no single maneuver can explore the complete *flight envelope* where the aircraft can safely fly. Hence the data obtained from one flight test maneuver can hopefully be used to estimate stability and control derivatives inside a small subset of the flight envelope. In this study, flight maneuvers follow the data provided by NASA Dryden Flight Research Facility.

Our presumed flight test data were obtained from the flight simulator with additive Gaussian noise. In the selected maneuver, the elevator deflection begins with -3.6 degrees followed by a doublet maneuver between -10 and 2 degrees (see Figure 10.6(a)). The responses of normal acceleration and angle of attack without noise are also depicted in time domain in Figures 10.6(b) and (c). These data are sampled every 0.05 second for a period of 5 seconds.

The application of our methodology to the realization of the lift coefficient C_L of the F-15 focused on the response of normal acceleration because of its strong correlation to this coefficient.

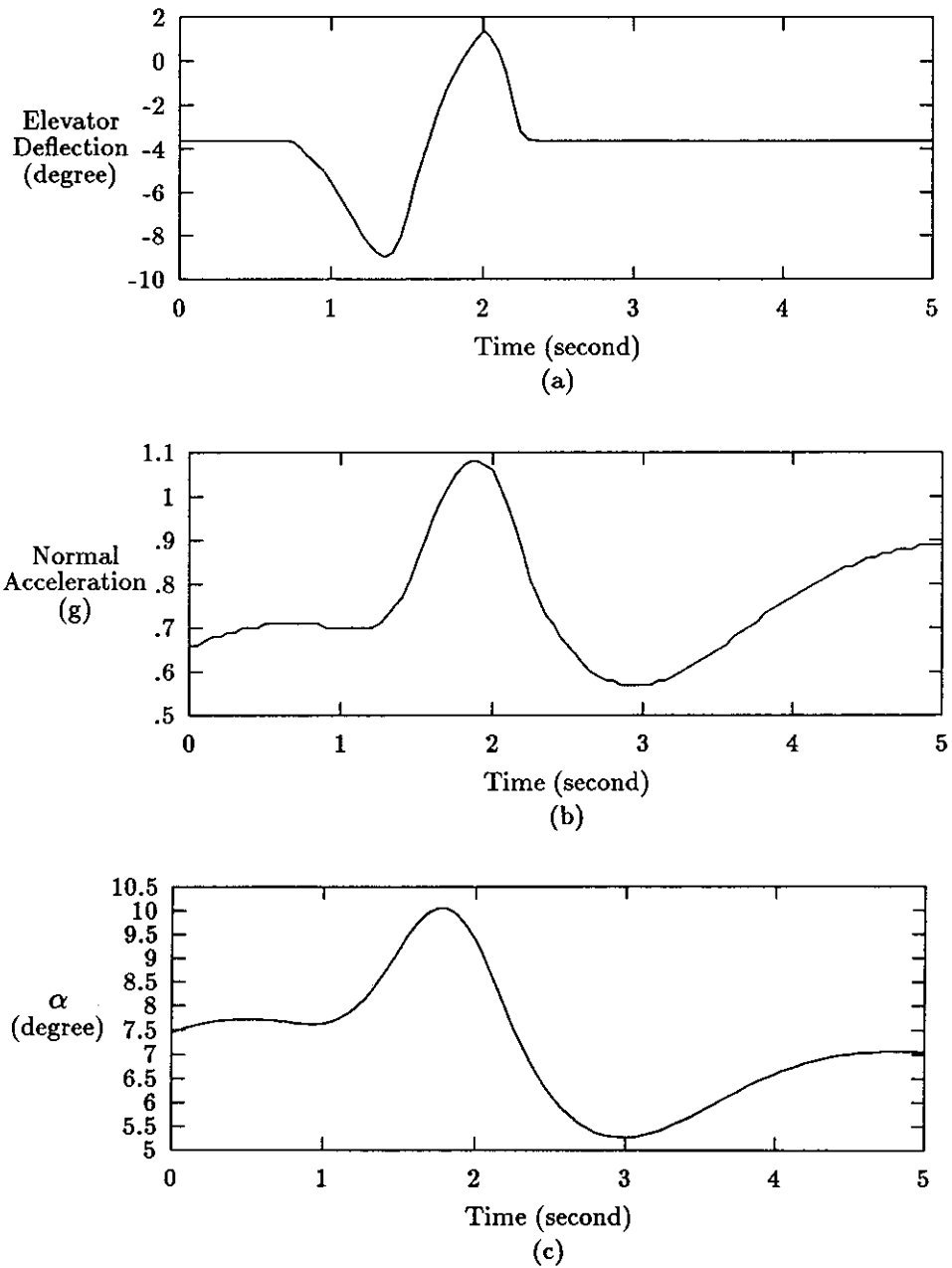


Figure 10.6: (a) The maneuver of elevator. (b) The response of normal acceleration. (c) The response of the angle of attack α .

Notice that the response of α varies between 4 and 12 degrees where the lift coefficient is only realizable under this experiment.

The lift coefficient C_L is at least a function of aircraft velocity M_c (V in terms of Mach number), elevator deflection δ_e , and angle of attack α . In the selected maneuver, M_c does not change much and can be considered as a constant effect on C_L . In addition, C_L is assumed to be linearly dependent on δ_e as implemented in the flight simulator. Therefore the nonlinear relation between C_L and α is the major concern of this study. The purpose of considering only this relation is to reduce the dimensionality of parameters and not to make the problem complicated at current stage. If we only consider α and δ_e as the independent variables of C_L , C_L is a three dimensional surface. In this study, the shape of this surface in α direction is unknown to be identified; while the shape in the δ_e direction just follows the flight simulator.

Since the realizable region of C_L is within its domain of $\alpha \in [4, 12]$ degrees and $\delta_e \in [-10, 2]$ degrees, C_L is assumed to be a surface bounded between the upper surface and the lower surface shown in Figure 10.7. The surface of any possible lift coefficient is assumed to be inside the range of $C_L \in [0.09, 0.34]$ at $(\alpha, \delta_e) = (4, -10)$, $C_L \in [0.185, 0.435]$ at $(\alpha, \delta_e) = (4, 0)$, $C_L \in [0.61, 0.96]$ at $(\alpha, \delta_e) = (12, -10)$, and $C_L \in [0.71, 1.06]$ at $(\alpha, \delta_e) = (12, 0)$. Figure 10.9 illustrates one of the 8 classes of functional regions obtained via the top down dichotomy algorithm which utilized 64 sample patterns for each class with systematic sampling scheme. This class contains the true lift coefficient that is actually the one implemented in the flight simulator.

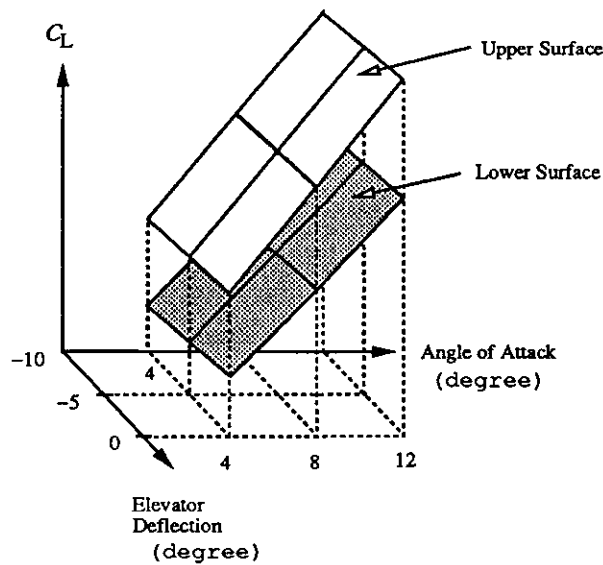


Figure 10.7: A possible functional region of the lift coefficient which is a surface between the upper surface and the lower surface.

10.6 Classification Results

The response of normal acceleration is the pattern vector of 100 components in this application. By Fisher's discriminant method, the pattern vector was transformed into a feature vector of 7 components. An adaptive neural network classifier with 3 exemplars per class was trained using 512 patterns (64 training patterns per class) and achieved a perfect classification for this training set.

Table 10.1 contains the classification results obtained in the presence of Gaussian noise whose standard deviation is set equal to its RMS amplitude. Every table entry of a classification result is the median of five simulations. Figure 10.8 shows the temporal responses of normal acceleration with and without noise. The noise-free response which is obtained from the original flight simulator is considered to be the system response observed from the true system without noise. 99% correct classification rate was obtained for 500 noisy versions of this response contaminated by random Gaussian noise with 0.0414 standard deviation and 0 mean, In

Table 10.1: Classification results for the training set with Gaussian noise.

standard deviation	0	0.0276	0.0414	0.0552	0.0828
SNR	∞	30	20	15	10
percent of correct classification	100	96.68	91.60	89.45	85.74

this case, the system response is classified with the class shown in Figure 10.9 whose functional region contains the true lift coefficient. This identified parameter space can be further partitioned to conduct hierarchical classifications.

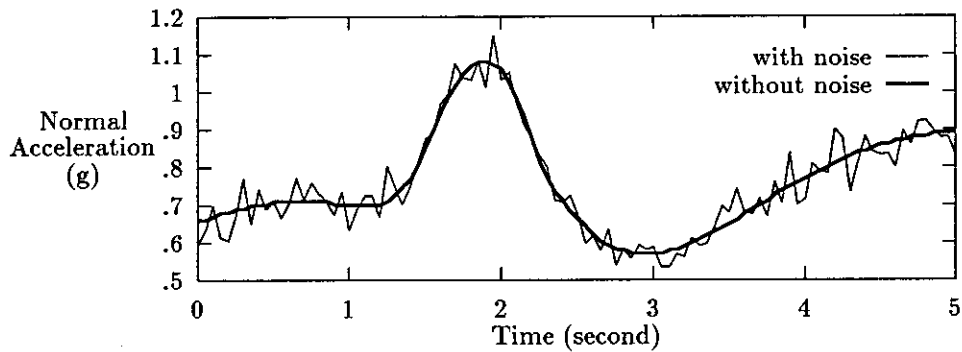


Figure 10.8: The response of normal acceleration in time domain corrupted by Gaussian noise with 0.0414 standard deviation and 0 mean.

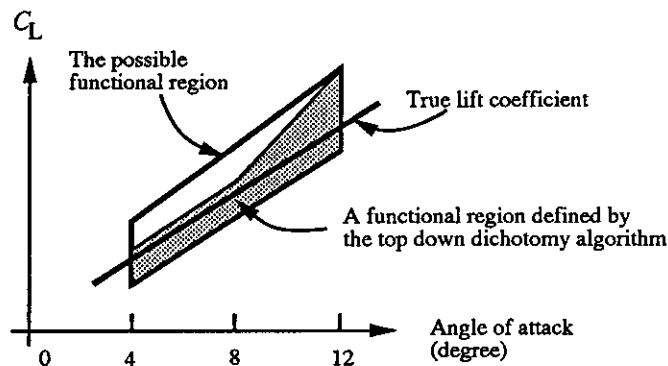


Figure 10.9: The functional region containing the true lift coefficient.

Applying the top down dichotomy algorithm to partition the parameter space

Table 10.2: Classification results for the training set of the second level partition with various levels of Gaussian noise.

standard deviation	0	0.0276	0.0414	0.0552	0.0828
SNR	∞	30	20	15	10
percent of correct classification	100	90.43	84.77	79.88	69.14

of the class containing the true lift coefficient, 8 subspaces corresponding to 8 pattern classes were obtained. The classification results for the training set of the second level partition corrupted by different levels of Gaussian noise are shown in Table 10.2. In the second level of hierarchical partition, the classification performance is more sensitive to noise than that in the first level because the parameter space is only 1/8 of the original parameter space.

10.7 Experiment Results on the SRV Flight Data

This section discusses the application of the proposed methodology to the real flight data of the powerless 3/8-scale F-15 airplane model (denoted as SRV). The SRV was designed to study recovery from stalls and spins. In the flight test, the SRV was launched from a modified B-52 airplane at an altitude of approximately 15,000 meters at a Mach number of 0.65. After the launch, the pilot on the ground flew the SRV remotely.

10.7.1 Flight Data

Figure 10.10 shows the responses and maneuver of the SRV in a time period of 5.6 seconds. These flight data are only a small segment of a flight test for about 5 minutes long when the SRV glided from about 15,000 meters to 5,000 meters above sea level. In the selected flight data, the maneuvers on the ailerons and rudder are

very small because only longitudinal motion is concerned here.

10.7.2 Objective

The objective is to utilize and modify the available F-15 simulator to characterize some unknown parameters of the SRV. In this experiment, the lift coefficient C_L , pitch moment coefficient C_m , and the drag coefficient C_D were investigated.

10.7.3 Experiment Design

As shown in Table 10.3, the aircraft dependent dimensions of the F-15 flight simulator are modified to be those of the SRV in order to generate a training set for the flight data of the SRV. The scaling factors for the length, area and volume are $\frac{3}{8} = 0.375$, $(\frac{3}{8})^2 = 0.140625$ and $(\frac{3}{8})^3 = 0.052734$, respectively. In following table, the data are obtained from the F-15 flight simulator and the SRV flight data.

Table 10.3: The scaling relationships between the F-15 and the SRV.

	F-15	SRV	scaling factor
surface area (ft ²)	608	86	0.1414
span (ft)	42.8	16.05	0.375
chord (ft)	15.95	5.98	0.375
weight (lb)	45000	2465	0.05478
moment of inertia IX	28700	275	0.009582
IY	165100	1808	0.01095
IZ	187900	2134	0.01136
product of inertia IXZ	-520	2.5	-0.0048

In this experiment the same maneuvers and initial states of the SRV flight were applied to the modified F-15 flight simulator with both engines shut off. In generating the training set, it is assumed that C_L , C_m , and C_D of the SRV also

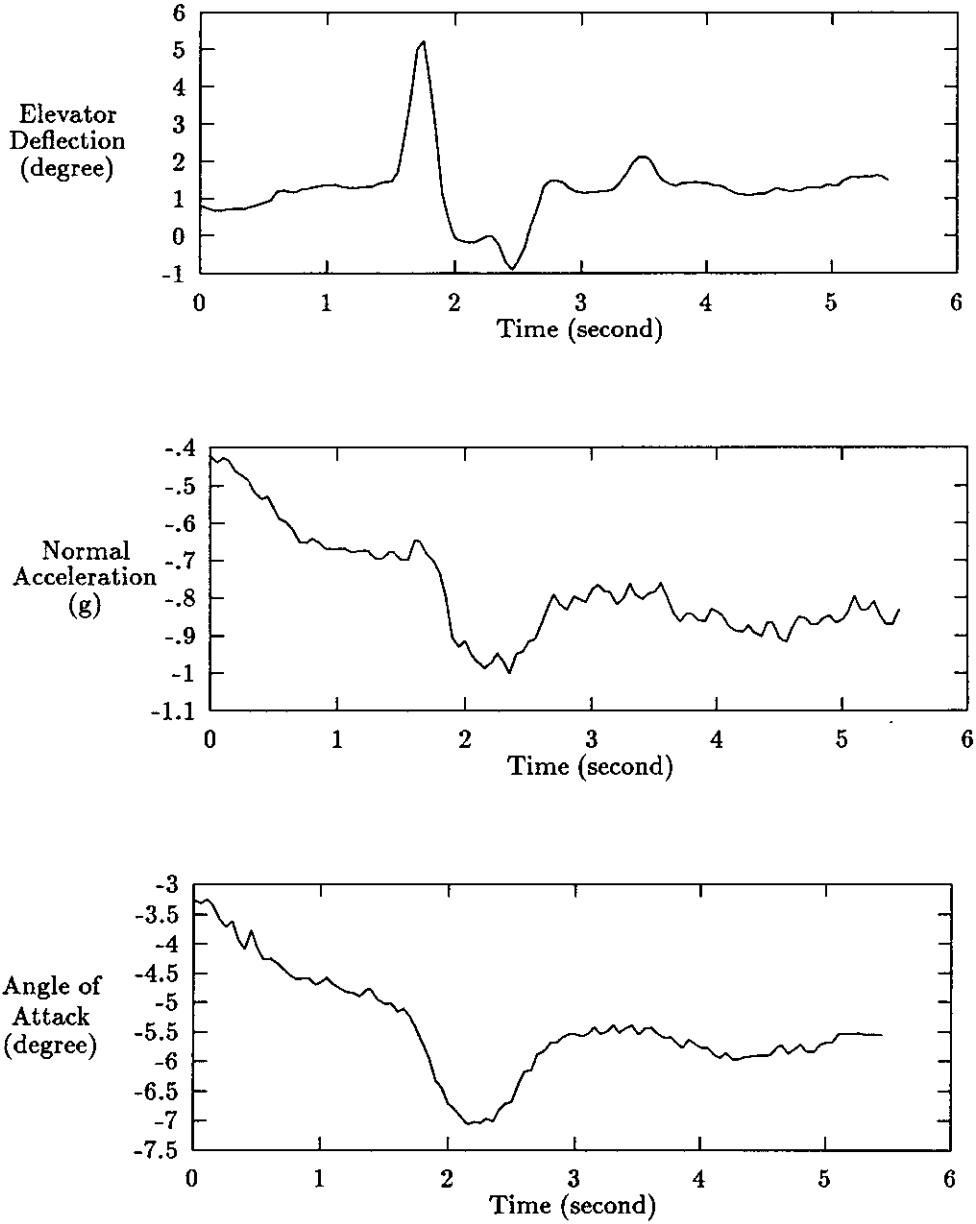


Figure 10.10: (a) The maneuver of elevator. (b) The response of normal acceleration. (c) The response of the angle of attack α .

have scaling relationships with respect to those parameters of the F-15. That is,

$$C_{L(SRV)} = a C_{L(F-15)}, \quad (10.7)$$

$$C_{m(SRV)} = b C_{m(F-15)}, \quad (10.8)$$

$$C_{D(SRV)} = c C_{D(F-15)}, \quad (10.9)$$

where a , b , and c are constants. This assumption is certainly not true. But this gives an approach to define a rough functional regions for these parameters of the SRV in terms of the parameters of the F-15, which are already available in the F-15 flight simulator. More precisely, the objective here is to identify functional regions for the unknown parameters in such a way that

$$C_{L(SRV)} = a C_{L(F-15)} \quad \text{with } a \in [a_1, a_2], \quad (10.10)$$

$$C_{m(SRV)} = b C_{m(F-15)} \quad \text{with } b \in [b_1, b_2], \quad (10.11)$$

$$C_{D(SRV)} = c C_{D(F-15)} \quad \text{with } c \in [c_1, c_2], \quad (10.12)$$

where a_1 , a_2 , b_1 , b_2 , c_1 , and c_2 are constants to be determined.

10.7.4 Experiment Results

The initial plausible functional regions confining the unknown parameters are assumed to be $a \in [0.4, 1.5]$, $b \in [0.4, 1.5]$, and $c \in [0.4, 1.5]$. Applying the top down dichotomy algorithm and using the observation of α as the system response, 6 disjoint functional regions were obtained. The flight data of the SRV were classified within the functional region of $a \in [0.95, 1.5]$, $b \in [0.4, 0.95]$, and $c \in [0.4, 1.5]$.

This identified functional region was then again partitioned into 9 disjoint functional regions. And the flight data were classified within the functional region of $a \in [1.225, 1.5]$, $b \in [0.4, 0.46875]$, and $c \in [0.4, 1.5]$ with an above 80% reliability of correct classification in the presence of noise with SNR of 15 (see Table 10.4).

Table 10.4: Percentage of correct classification results for the training set and the SRV flight data of the second level partition with various levels of Gaussian noise.

standard deviation	0	0.183	0.274	0.365	0.548
SNR	∞	30	20	15	10
training set	100	91.84	86.11	82.29	73.78
SRV flight data	100	95	87.09	80.88	72.31

It is obvious that the system response of α is very sensitive to the variation of C_m , while less sensitive to the variation of C_D . Table 10.4 presents the estimated overall classification performance and the confidence levels of classification results for the SRV flight data with various levels of Gaussian noise.

10.8 Discussion and Conclusions

The proposed methodology was successfully applied to the characterization of the lift coefficient of the F-15 airplane. The classification results show an 80 percent correct classification in the presence of noise with an overall SNR of 15 though the experiment data were obtained analytically from the simulation with additive Gaussian noise. A hierarchical classification technique was employed to refine the identified functional region for the lift coefficient. Applying the region growing technique mentioned in section 7.3, the reliability of classification results should be improved to make the results more useful. In this case, the testing data were obtained from the flight simulator, and the simulated lift coefficients were already known. Therefore, the classification results can be validated to show a high percentage of correctness, like the results obtained in Chapter 9. These results also demonstrate the applicability and reliability of the proposed methodology.

This methodology was also applied to the flight data of the 3/8-scale F-15 airplane model to characterize its lift coefficient, pitch moment coefficient, and

drag coefficient. The results obtained were the functional regions for the unknown parameters in terms of those parameters of the F-15 flight simulator. That is,

$$C_{L(SRV)} \in [1.225 \cdot C_{L(F-15)}, 1.5 \cdot C_{L(F-15)}], \quad (10.13)$$

$$C_{m(SRV)} \in [0.4 \cdot C_{m(F-15)}, 0.46875 \cdot C_{m(F-15)}], \quad (10.14)$$

$$C_{D(SRV)} \in [0.4 \cdot C_{D(F-15)}, 1.5 \cdot C_{D(F-15)}]. \quad (10.15)$$

The lift coefficient of the SRV is confined within the two surfaces defined in terms of the lift coefficient of the F-15 flight simulator, so are the pitch moment coefficient and the drag coefficient. This information can be used to construct more appropriate model structures for these coefficients when applying optimization techniques. Since the true answers for these coefficients of the SRV are not available, it is difficult to verify the identified results. However, the results seem to be reasonable based on the identified functional regions and the implications from previous case studies.

In the experiment on the flight data of the SRV, only the longitudinal motion was examined. Small perturbations of pilot inputs are presented in the lateral motion of the selected section of the flight data. The lift, pitch moment, and drag coefficients are assumed to affect the longitudinal motion greatly, and the other aerodynamic coefficients are assumed to be equal to those of the F-15 flight simulator. The assumptions might not be good. Therefore the performance of the identified results may be degenerated by the defective assumptions. The coupling effect of all the aerodynamic coefficients on both longitudinal and lateral motions requires further investigations. Currently, the proposed approach is limited to a low dimensional parameter space and is not practical to consider too many parameters at one time.

CHAPTER 11

Conclusions

11.1 Summary

The problem of obtaining mathematical models of physical dynamic systems from noisy observations is system identification, which is of great importance, not only in the traditional engineering disciplines, but also in biological systems, economic science, or social science. In this dissertation, a novel approach based on pattern recognition techniques is proposed to attack this problem.

The pattern recognition techniques developed in this research are employed to characterize the nonlinear parameters of the differential equations that govern the dynamic system being modeled. A problem solving architecture including an off-line training phase and an on-line identification phase is proposed. The training phase consists of:

1. *pattern designation,*
2. *feature selection,*
3. *classifier implementation,* and
4. *performance evaluation.*

The identification phase includes:

1. *feature extraction* and
2. *pattern classification*

11.1.1 Pattern Designation

Pattern designation formulates parameters characterization as a pattern recognition problem. Two techniques (piecewise linear functions and piecewise smooth polynomials) are proposed to approximate the unknown parameters of a real world system. This functional approximation parameterizes the models for the unknown parameters (or subsystems of a complex system) and also defines a parameter space representing candidate models. Every candidate model is designated to be a “pattern.” A top down dichotomy algorithm then partitions this parameter space into disjoint regions such that models in any one region generate system responses within a certain variation threshold. Every disjoint region in the parameter space corresponds to a pattern class. This algorithm recursively partitions the parameter space based on the variation of simulated system responses of models sampled in the parameter space in order to maintain the coherence property of any pattern class.

11.1.2 Feature Selection

The feature extraction scheme is a linear transformation which minimizes the variation of features within any one class while maximizing the variation of features between classes. This transformation fulfills the objectives to increase the clustering within any one pattern class, to enhance the difference between pattern classes, and to reduce the data dimensionality. The transformation matrix is composed of the eigenvectors corresponding to the largest eigenvalues of the matrix $S_w^{-1}S_b$ where S_w and S_b are the estimated within-class and between-class covariance matrices of the system responses, respectively. Features are selected from the transformed system response using the transformation matrix. In addition, *ad hoc* approaches, such as power spectra and exponential component analysis techniques,

may serve to extract nonlinear features.

11.1.3 Classifier Implementation

An adaptive neural network classifier is trained to memorize exemplars of every pattern class. Its learning algorithm is based on a modification of the Learning Vector Quantizer (LVQ), which distributes exemplars into decision regions in such a way that every exemplar dominates a region corresponding to its representing pattern class. The modified learning algorithm developed in this dissertation defines much more precise decision boundaries and outperforms LVQ. This neural network can be implemented in an architecture similar to the Hamming net to utilize the massive parallelism.

11.1.4 Performance Evaluation

Finally, performance evaluation is conducted by estimating the misclassification rates and the noise effect on classifications. The approach to analyze the overall classification performance and the reliability of a pattern classification is developed. Since the top down dichotomy algorithm provides a fairly even distribution of the sampled training patterns in the system response space, the classification performance estimated from the training set is reliable.

The overall classification performance gives an indication of the efficiency of the problem solving procedure from pattern designation, to feature selection, to classifier implementation. The procedure can be repeated if the classification performance is not satisfactory.

The identified result of a pattern classification is associated with a percentage of correct classification which provides a measure of confidence in the realization of the mathematical descriptors of the unknown nonlinear parameters. Both region

growing and region merging techniques are proposed to improve the reliability of classification results.

11.1.5 On-line Identification

In the identification phase, the actual system response is taken as an input to the pattern recognition system obtained in the training phase. The real world system is classified with the model class of the most similar system response. Since the pattern recognition system is massively parallel, system characterization is accomplished instantaneously.

The pattern recognition system has learned the patterns in the plausible parameter space, and the neural network classifier has memorized the typical responses of these patterns. In a real world, the characteristics of a system component may change dramatically because of aging and heating processes. It is very effective and efficient to utilize this pattern recognition system to classify the changing characteristics rapidly.

The off-line training process may require highly substantial computations. But, the algorithms developed in the research are well suited for massively parallel implementation in both the neural network training and automatic pattern class designation.

The methodology was successfully applied to the realization of the characteristics of the critical components or parameters of two real world systems, the Space Station Freedom and the 3/8-scale F-15 airplane model. The results are encouraging, showing a high percentage of correct classification in a noisy environment. The identification results of the parameter characteristics are thus very reliable and useful for further parameter estimation.

11.2 Limitations

According to the sampling schemes employed in this dissertation, the number of sample patterns becomes combinatorial explosive if the dimensionality of the parameter space gets larger. The proposed approach becomes computationally intractable and impractical in the off-line training phase. Currently, this approach is thus not scalable and limited to a low dimensional parameter space.

11.3 Implications

There are several implications stemmed from the case studies described in Chapters 8, 9, and 10:

1. The pattern recognition system has explored the plausible functional regions for the unknown nonlinear parameters and is capable of detecting and identifying the changing characteristics of the parameters rapidly.
2. In the experiment on the SRV flight data, small perturbations of maneuvers are required for the local linearity assumption of parameters when the optimization approaches are used. The approach proposed in this dissertation deals with nonlinear parameters and is able to handle flight data with a larger range of maneuvers.

11.4 Applications

Based on the limitations and implications discussed above, the proposed approach can be applied to systems with

- several critical nonlinear system parameters needed to be modeled and

- the parameter characteristics vary in a wide range during system operations, especially due to heating or aging processes.

11.5 Contributions

A methodology based on pattern recognition techniques has been developed to identify functional regions for the unknown parameters of a physical system. These functional regions confining the unknown parameters provide useful information in selecting initial models for the parameters when the search for the best model is conducted. The specific contributions of the research described in this dissertation are summarized as follows:

1. Formulation of the system characterization problem to be a pattern recognition problem and implementation of a pattern recognition system. To construct the pattern recognition system, the following novel ideas are introduced:
 - A top down dichotomy algorithm, described in section 4.4, which automates the pattern class designation,
 - An improvement on the Fisher's discriminant method, described in section 5.5, for making the feature extractor robust to noise,
 - An adaptive learning rule, described in section 6.3.2, for constructing the neural network classifier which performs better than conventional classifiers, like the minimum distance classifier.
2. Establishment of a problem solving paradigm combining unsupervised and supervised learning processes. Unsupervised learning automatically partitions the problem domain based on the problem nature, while supervised

learning promotes classification performance. Actually, the top down dichotomy algorithm is an unsupervised learning process, and the MLVQ learning rule is a supervised learning process.

This research intends to study a complement of conventional system identification approaches which often employ the optimization techniques and fail in many situations. The approach developed here can provide useful information for the optimization approaches with better starting models in order to reduce search steps and to avoid local minima. The case studies demonstrate the applicability of the proposed approach.

11.6 Suggestions for Future Research

A new and improved methodology is presented to characterize the nonlinear parameters of a physical dynamic system. Its performance and limitations are also discussed. The following areas may require further investigations in order to improve the performance and applicability of this methodology.

11.6.1 Syntactic Pattern Recognition Approach

The pattern recognition method developed in this dissertation is a statistical approach which extracts and classifies features of patterns based on the statistical significance of a training set. If engineering insight to the system responses is available, a syntactic pattern recognition approach can be applied to the system identification problem. In other words, some specific signals in the system responses may associate with physical interpretations which characterize the system. The primitive features of the system responses may still need statistical approach to identify. However, syntactic grammars can be used to describe the interrelationships between these primitive features. Thus, the pattern recognition

system may come up with the combination of a neural network and an expert system. The neural network detects the low level features; while the expert system utilizes these features to infer and to characterize the system being modeled.

11.6.2 Reduction of the Sample Size

Although the algorithms developed in this dissertation are well suited for parallel implementation, the large sample size of sample patterns in the parameter space requires substantial amount of computation. The only information utilized by the top down dichotomy algorithm is the variation among the sample pattern vectors. If other useful information can be extracted from these sample pattern vectors, the sample size may be reduced to improve the efficiency. The useful information includes higher order statistics and the relationships between neighboring patterns.

11.6.3 Other Applications

The problem solving methodology which combines the unsupervised and supervised learning techniques can be applied to other applications, like control problems. Suppose that the objective is to design the control strategies for a complex system whose mathematical model is not available. Also, the behavior of this complex system is not well understood. The unsupervised learning technique or the top down dichotomy algorithm can be employed to partition the state space of this system; and then the supervised learning technique is applied to assign control strategies for every partitioned subspace.

APPENDIX A

Splitting Dimension Analysis

This appendix is concerned with the selection of the splitting dimension of the parameter space in the top down dichotomy algorithm. Let us consider the scatter indication of pattern vector X_i 's inside the current region C of the parameter space

$$s = \sum_{i \in C} (X_i - M)^T (X_i - M) \quad (\text{A.1})$$

and the variance indication

$$\sigma = \frac{s}{n} \quad (\text{A.2})$$

where M is the mean pattern vector of all patterns in C and n is the number of patterns in C . The scalar s is the trace of the scatter matrix of X_i 's. According to the top down dichotomy algorithm, σ is a variation measurement of patterns sampled from C .

If the splitting dimension is chosen to be dimension j , the scatter indications of the two disjoint subregions C_1^j and C_2^j after splitting are

$$s_1^j = \sum_{i \in C_1^j} (X_i - M_1^j)^T (X_i - M_1^j) \quad (\text{A.3})$$

and

$$s_2^j = \sum_{i \in C_2^j} (X_i - M_2^j)^T (X_i - M_2^j) \quad (\text{A.4})$$

where M_1^j and M_2^j are the mean pattern vectors of patterns in C_1^j and C_2^j , respectively.

Because of systematic sampling or stratified sampling with equal size of strata, the number of patterns in C_1^j and C_2^j is both equal to $\frac{n}{2}$. Thus $M = \frac{1}{2}(M_1^j + M_2^j)$.

It can be derived from Equation (A.1)

$$\begin{aligned}
s &= \sum_{i \in C_1^j} (X_i - M)^T (X_i - M) + \sum_{i \in C_2^j} (X_i - M)^T (X_i - M) \\
&= \sum_{i \in C_1^j} [(X_i - M_1^j)^T (X_i - M_1^j) - (X_i - M_1^j)^T (M_2^j - M_1^j) + \frac{1}{4} (M_2^j - M_1^j)^T (M_2^j - M_1^j)] + \\
&\quad \sum_{i \in C_2^j} [(X_i - M_2^j)^T (X_i - M_2^j) - (X_i - M_2^j)^T (M_1^j - M_2^j) + \frac{1}{4} (M_1^j - M_2^j)^T (M_1^j - M_2^j)] \\
&= s_1^j + s_2^j + \frac{n}{4} (M_1^j - M_2^j)^T (M_1^j - M_2^j). \tag{A.5}
\end{aligned}$$

The term $(M_1^j - M_2^j)^T (M_1^j - M_2^j)$ is actually the Euclidean distance between M_1^j and M_2^j .

Therefore, the selection of dimension j that fulfills

$$\max_j \{(M_1^j - M_2^j)^T (M_1^j - M_2^j)\} \tag{A.6}$$

is equivalent to that of

$$\min_j \{s_1^j + s_2^j\}. \tag{A.7}$$

This shows the reason for the selection of splitting dimension in order to reduce the scattering and to stop partitioning as early as possible.

APPENDIX B

Generalized Eigenvalue Problem

The *generalized eigenvalue problem* is formulated as follows:

$$\Sigma_1 x = \lambda \Sigma_2 x. \quad (\text{B.1})$$

When Σ_2 is nonsingular, Equation (B.1) is equivalent to the ordinary eigenvalue problem

$$(\Sigma_2^{-1} \Sigma_1) x = \lambda x. \quad (\text{B.2})$$

Suppose that Σ_1 and Σ_2 are symmetric matrices and Σ_2 is positive definite. This problem can be handled by the ordinary eigenvalue problem technique. First of all, Σ_2 is diagonalized by a orthonormal transformation because of its symmetry and positive definite properties. That is,

$$\Sigma_2 = \Phi \Lambda \Phi^T = G G^T, \quad (\text{B.3})$$

where Λ is a diagonal matrix, $G = \Phi \Lambda^{\frac{1}{2}}$, and $G^T = \Lambda^{\frac{1}{2}} \Phi^T$. Then Equation (B.1) yields

$$\Sigma_1 x = \lambda G G^T x. \quad (\text{B.4})$$

Multiplying G^{-1} to the left of both sides and applying the property that $G^{-T} G^T = I$, we can get

$$G^{-1} \Sigma_1 G^{-T} G^T x = \lambda G^T x. \quad (\text{B.5})$$

Let $\Sigma = G^{-1} \Sigma_1 G^{-T}$, then Equation (B.1) is transformed into the ordinary eigenvalue problem:

$$\Sigma v = \lambda v \quad (\text{B.6})$$

where v and λ are respectively a pair of eigenvector and eigenvalue of matrix Σ . Since Σ_1 is symmetric, Σ is also symmetric. The ordinary eigensystem technique, such as Jacobi's method [PFTV88], in solving the eigensystem of a symmetric matrix can be applied to Σ . Once the eigensystem of Σ is solved,

$$x = G^{-T}v = \Phi\Lambda^{-\frac{1}{2}}v. \quad (\text{B.7})$$

Bibliography

- [AE71] K. J. Astrom and P. Eykhoff. System identification – a survey. *Automatica*, 7:123–162, 1971.
- [AK82] Kofi Apenyo and Walter J. Karplus. A domain segmentation approach to modeling distributed parameter systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-12(3):299–307, May/June 1982.
- [Bek70] George A. Bekey. System identification – an introduction and a survey. *Simulation*, pages 151–166, October 1970.
- [Bri74] E.O. Brigham. *The Fast Fourier Transform*. Prentice-Hall, Englewood Cliffs, New Jersey, 1974.
- [CG86] G.A. Carpenter and S. Grossberg. Neural dynamics of category learning and recognition : attention, memory consolidation and amnesia. In J. Davis, R. Newburgh, and E. Wegman, editors, *Brain Structure, Learning, and Memory, AAAS Symposium Series*. 1986.
- [Che73] C-H Chen. *Statistical Pattern Recognition*. Spartan Books, Hayden Book Company, Inc., Rochelle Park, New Jersey, 1973.
- [Coc77] William G. Cochran. *Sampling Techniques*. John Wiley & Sons, New York, 3rd edition, 1977.
- [DAK88] Eugene L. Duke, Robert F. Antoniewicz, and Keith D. Krambeer. Deviation and definition of a linear aircraft model. Technical Report NASA RP-1207, NASA, 1988.
- [DH73] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, 1973.
- [Etk82] Bernard Etkin. *Dynamics of Flight: Stability and Control*. John Wiley & Sons, New York, 2nd edition, 1982.
- [Fis36] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Ann. Eugenics*, 7:179–188, 1936.
- [FMI83] K. Fukushima, S. Miyake, and T. Ito. Neocognitron: a neural network model for a mechanism of visual pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13:826–834, 1983.

- [Fu82] K.S. Fu. *Syntactic Pattern Recognition and Applications*. Prentice-Hall, Englewood Cliff, New Jersey, 1982.
- [Fuk72] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, New York, 1972.
- [Fuk90] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, New York, 2nd edition, 1990.
- [Gol86] B. Gold. Hopfield model applied to vowel and consonant discrimination. In *Proceedings of the Neural Networks for Computing, Snowbird, Utah*, pages 158–164, April 1986.
- [GS86] P. M. Grant and J. P. Sage. A comparison of neural network and matched filter processing for detecting lines in images. In *Proceeding of the Neural Networks for Computing, Snowbird, Utah*, pages 194–199, April 1986.
- [GS91] Shiomu Geva and Joaquin Sitte. Adaptive nearest neighbor pattern classification. *IEEE Transactions on Neural Networks*, 2(2):318–322, March 1991.
- [Heb49] D. Hebb. *The Organization of Behaviour*. Wiley, New York, 1949.
- [Hop82] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *Proc. Natl. Acad. Sci. USA*, pages 2554–2558, April 1982.
- [Hop84] J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Natl. Acad. Sci. USA*, 81:3088–3092, May 1984.
- [HT85] J. J. Hopfield and D. W. Tank. “Neural” computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152, 1985.
- [Kar72] Walter J. Karplus. System identification and simulation – a pattern recognition approach. In *AFIPS – Conference Proceedings*, volume 41, pages 385–392, 1972.
- [Koh84] Teuvo Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, New York, 1984.
- [Lac75] Peter A. Lachenbruch. *Discriminant Analysis*. Hafner Press, New York, 1975.
- [Lev69] M.D. Levine. Feature extraction: A survey. *Proc. IEEE*, 57:1391–1407, 1969.

- [LGM87] R.P. Lippmann, B. Gold, and M.L. Malpass. A comparison of Hamming and Hopfield neural nets for pattern recognition. Technical Report 769, Lincoln Laboratory, MIT, 1987.
- [Lip87] Richard P. Lippmann. An introduction to computing with neural nets. *IEEE ASSP Magazine*, pages 4–22, April 1987.
- [Lju87] Lennart Ljung. *System Identification – Theory for the User*. Prentice-Hall, Englewood Cliffs, New Jersey, 1987.
- [Mat80] Bertil Matérn. *Spatial Variation*. Springer-Verlag, New York, 2nd edition, 1980.
- [MP43] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, pages 115–133, 1943.
- [MP69] Marvin Minsky and Seymour Papert. *Perceptrons*. MIT Press, Cambridge, MA, 1969.
- [OE78] Robert K. Otnes and Loren Enochson. *Applied Time Series Analysis*, volume 1. John Wiley & Sons, New York, 1978.
- [PFTV88] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, Cambridge, New York, 1988.
- [Pro76] S. W. Provencher. A Fourier method for the analysis of exponential decay curves. *Biophysical Journal*, 16:27–41, 1976.
- [RM86] D. E. Rumelhart and J. L. McClelland. *Parallel Distributed Processing, Vol. I*. Bradford Books/MIT Press, Cambridge MA, 1986.
- [Ros58] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.
- [Ros62] F. Rosenblatt. *Principles of Neurodynamics*. Spartan, New York, 1962.
- [SCS76] M. R. Smith and S. Cohn-Sfetcu. Decomposition of multicomponent exponential decays by spectralanalytic techniques. *Technometrics*, 18(4):467–482, November 1976.
- [Sed90] Robert Sedgewick. *Algorithms in C*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1990.

- [SH74] G. N. Saridis and R. F. Hofstadter. A PR approach to the classification of nonlinear systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-4, 1974.
- [Sim75] T.M. Simundich. *System Characterization: A Pattern Recognition Approach*. PhD thesis, University of California, Los Angeles, 1975.
- [Sim88] S. S. Simonian. On-orbit system parameter identification. In *Workshop on Modal Determination for Large Space System, JPL D-5574*, volume 1, March 1988.
- [SK86] Yoshitaka Shibata and Walter J. Karplus. Distributed source identification in river pollution systems by pattern recognition. *Transactions of The Society for Computer Simulation*, 3(1):1-29, January 1986.
- [SR86] T. J. Sejnowski and C. Rosenberg. Nettek: a parallel network that learns to read aloud. Technical Report JHU/EECS-86/01, The Johns Hopkins University Electrical Engineering and Computer Science, 1986.
- [TH86] D. W. Tank and J. J. Hopfield. Simple "neural" optimization networks: an A/D converter, signal decision circuit, and a linear programming circuit. *IEEE Transactions on Circuits and Systems*, CAS-33(5):533-541, May 1986.
- [WH60] B. Widrow and M.E. Hoff. Adaptive switching circuits. *1960 IRE WESCON Convention Record, Part 4*, pages 96-104, August 1960.