AN INFORMATION-THEORETIC APPROACH TO SONAR AUTOMATIC TARGET RECOGNITION

By

RODNEY ALBERTO MOREJON

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2003

To the Memory of My Father

ACKNOWLEDGMENTS

The quest of pursuing a doctoral degree is often a long and arduous journey requiring enduring patience, perseverance, and personal sacrifice. It is not only the pursuer of the degree who requires this dedication and sacrifice, but also those who are close to him. I would like to express my extreme gratitude to those who have helped me throughout this extended journey.

First of all, I would like to thank my advisor, Dr. Jose Principe, for his patience, support, and guidance over my long course of study (which has extended beyond my longest expectations when I began nearly 10 years ago). I am also grateful to all the members of my advisory committee. I am especially thankful of Dr. Gerry Dobeck, who has served as a second mentor through the course of my research.

Additionally, I cannot express my level of gratitude to Mr. Kirk Dye (without whom the opportunity to pursue this degree not just once, but twice, would have never been possible). I am also grateful to the Coastal Systems Station and Dr. Dave Skinner, whose support in providing both financial and temporal resources has been invaluable.

Lastly, I am eternally grateful to my family, especially my parents and grandparents, without whom I would have never been exposed to the many opportunities I have enjoyed throughout my life. And finally, I would like to thank my daughter, Maria, who has become a welcome inspiration in my life. Her patience, even at age two, in allowing Daddy to finish his schoolwork despite the temptation of dozens of playtime activities has been remarkably cool. Now is the time to play.

TABLE OF CONTENTS

page

ACKNOWLEDGMENTS iv

LIST OF TABLES ix

LIST OF FIGURES x

ABSTRACT xiv

CHAPTER

1 INTRODUCTION 1

Background 2
Motivation 5
Overview of Dissertation 7

2 SONAR AUTOMATIC TARGET RECOGNITION (ATR) 8

Introduction 8
Underwater Sonar Image Characterization 8
General Multistage ATR 10
Sonar Imagery ATR Methods 12
Preprocessing/Image Normalization Approaches 12
Detection Approaches 13
Feature Extraction Approaches 14
Classification Approaches 16
Fusion Approaches 16
Feature Extraction Overview 17
Feature Selection 20
Principal Component Analysis (PCA) 21
Independent Component Analysis (ICA) 22
Linear Discriminant Analysis (LDA) 22
Other Common Methods 23
Sonar Datasets 23
Sonar Set 10 (SON10) 23
Sonar Set 5 and 12 (SON512) 25
Sonar Baseline Extracted Feature Sets 26

3 INFORMATION-THEORETIC LEARNING (ITL) 28

Introduction 28
Information Theory 28
General Learning System Architecture 31
Information Theoretic Optimization Criterion 32
Information Theoretic Learning Approach 34

4 ALGORITHM OPTIMIZATION 39

Introduction 39
Baseline Entropy Algorithm 39
Entropy Algorithm Factorization 40
Exploitation of Symmetry 42
Entropy Vectorization 43
Results for the QMI Criterion 45
Conclusions 46

5 ADVANCED PARAMETER SEARCH TECHNIQUES 48

Introduction 48
Baseline Technique: Gradient Descent (GD) 48
Advanced Search Techniques 49
Improved GD Algorithms 49
Resilient Backpropagation (RP) 50
Conjugate Gradient Algorithms 51
Newton's Method 53
Levenberg-Marquardt (LM) Algorithm 54
Performance Divergence Measures 54
Line Search Routines 55
Difficulties and Remedies for ITL 56
Adaptive Kernel Size 56
Positive and Negative Performance Values 58
Relative Error versus Absolute Squared-Error 60
Summary of Advanced Parameter Search Algorithms for ITL 63
Sample Problems 64
Example 1: Maximum Entropy Feature Extraction 65
Example 2: Frequency Doubler 68
Example 3: Simplified Frequency Doubler 71
Conclusions 75

6 BATCH TRAINING APPROACH 77

Introduction 77
Batch Training Approach 77
Overview 77
Batch Training Parameters 79

Batch Selection Methods 81
Other Batch Training Considerations 81
Stochastic Information Gradient (SIG) Method 82
Batch Training Comparison Results 82
Performance Comparison Metric 82
Variations of the SIG 83
Batch Training with Advanced Parameter Adaptation 83
Batch Selection Methods 85
Batch Training Comparison with Additive Noise 88
Batch Training Comparison with Oversampling 90
Discussion on Batch Parameter Selection 92
Conclusions 94

7 PROPERTIES OF THE INFORMATION ESTIMATORS 96

Introduction 96
Entropy Estimator 96
Families of Entropy Curves 97
Kernel Size Ranges 102
Effects of Different Scale and a Scale-Invariant Entropy 104
Sample Size Variations, Relative Entropy and Batch Size Estimation 107
Entropy-Based Intrinsic Dimensionality Estimators 112
Relative entropy intrinsic dimensionality estimator 113
Information potential dimensionality estimator 114
Dimensionality estimation examples 118
Mutual Information Estimators 120
Families of Mutual Information Curves 120
Effects of Sample Size Variations 122
Effects of Different Scale 125
Effects of Dimensional Differences 127
Conclusions 133

8 SONAR FEATURE EXTRACTION 135

Introduction 135
Informative Feature Mapping 136
Approach 136
Results 138
Feature Subset Ranking and Selection 143
Approach 143
Results 149
Discriminative Feature Mapping 154
Approach 154
Results 155
Summary of Results 165
Conclusions 167

9 CONCLUSIONS.....169

APPENDIX IMPLEMENTATION DETAILS FOR TRAINING ALGORITHMS.....172

Gradient Descent for ITL..... 172
Levenberg-Marquardt with Line-Search for ITL..... 172

REFERENCE LIST 174

BIOGRAPHICAL SKETCH 178

LIST OF TABLES

Table		page
2-1	Distribution of extracted feature exemplars for SON10.	25
2-2	Distribution of extracted feature exemplars for SON512.	25
2-3	The CSS baseline extracted feature set.	27
4-1	Algorithm optimization: Factorization.	41
4-2	Algorithm optimization: Symmetry with factorization.	43
4-3	Algorithm optimization: Vectorized with symmetry and factorization.	44
4-4	Algorithm optimization: ED-QMI baseline and optimized results.	46
5-1	Advanced parameter search algorithm modification summary for ITL.	64
6-1	Batch training parameter settings.	84
7-1	Comparison of dimensionality estimators.	118
8-1	Exhaustive search feature sets and times for SON10 and SON512.	145
8-2	Forward/backward search feature sets for SON10 and SON512.	149
8-3	Summary of detection performance for SON10.	165
8-4	Summary of detection performance for SON512.	166

LIST OF FIGURES

Figure		page
2-1	Representative sonar image.	9
2-2	Multistaged ATR approach.	11
2-3	Normalized sonar image.	13
2-4	Typical SON10 sonar image.	24
2-5	Typical SON512 sonar image.	26
3-1	General learning system architecture.	31
4-1	Entropy estimator optimization results.	45
5-1	Performance surface variation with kernel size.	57
5-2	Performance divergence measure.	59
5-3	Levenberg-Marquardt ITL algorithms: Weight tracks.	62
5-4	Levenberg-Marquardt ITL algorithms: Training efficiency.	63
5-5	Maximum entropy feature extraction.	66
5-6	Feature extraction: Parameter search comparison in epochs.	66
5-7	Feature extraction: Parameter search comparison in FLOPs.	67
5-8	Frequency doubler: Input and desired signals	68
5-9	Frequency doubler: Network topology	69
5-10	Frequency doubler: Local minima outputs.	70
5-11	Frequency doubler: Parameter search comparison.	70
5-12	Desired and ED-QMI optimal signals.	72

5-13 Fixed kernel size weight tracks. 73
5-14 Fixed and adaptive kernel size weight tracks for IC #3. 74
5-15 Kernel size and training performance for RP and IC#3. 75
6-1 Batch training comparison for GD. 84
6-2 Batch training comparison for SCG. 85
6-3 Batch training comparison for RP. 86
6-4 Batch training comparison for LM. 86
6-5 Batch training selection method comparison with GD. 87
6-6 Batch training selection method comparison with SCG. 87
6-7 Original and noise corrupted signals. 89
6-8 Batch training comparison: GD with noise. 89
6-9 Batch training selection method comparison: GD and noise. 90
6-10 Batch training comparison: GD with oversampled. 91
6-11 Batch training selection method comparison: GD with Oversampled. 92
7-1 Family of entropy curves: Variable scale. 98
7-2 Family of entropy curves: Variable sample size. 99
7-3 Family of entropy curves: Variable dimensionality. 99
7-4 Scale translation examples. 105
7-5 Family of entropy curves: Kernel size equivalents. 108
7-6 Relative entropy comparison across variable sample sizes. 110
7-7 Relative entropy comparison for sample problems. 111
7-8 Relative entropy comparison for SON512. 111
7-9 Relative entropy ID estimation for SON512 training set. 115
7-10 The PCA residual errors for SON512 training set. 115
7-11 Dimension-normalized entropy curves. 116

7-12 Grassberger-Procaccia and IP dimension estimators. 117
7-13 Dimension estimation dependency on sample size. 119
7-14 Scatter plot for mutual information experimentation data. 121
7-15 Family of CS-QMI curves: Variable sample size. 122
7-16 Family of ED-QMI curves: Variable sample size. 123
7-17 Family of QMI information force curves: Variable sample size. 125
7-18 Family of CS-QMI curves: Variable scale ratios. 126
7-19 Family of ED-QMI curves: Variable scale ratios. 127
7-20 Family of CS-QMI curves: Variable equal dimensionality. 128
7-21 Family of ED-QMI curves: Variable equal dimensionality. 128
7-22 Family of CS-QMI curves: Variable different dimensionality. 130
7-23 Family of ED-QMI curves: Variable different dimensionality. 130
7-24 Family of CS-QMI curves: Variable dimensionality compensated. 132
7-25 Family of ED-QMI curves: Variable dimensionality compensated. 132
8-1 Maximum entropy informative feature extraction for dimension reduction. 138
8-2 Classification performance of MaxEnt and PCA features for SON10. 140
8-3 Classification performance of MaxEnt and PCA features for SON512. 141
8-4 Comparison of relative entropy of MaxEnt and PCA features. 142
8-5 Maximum QMI feature ranker. 144
8-6 Individual feature CS-QMI for SON10 and SON512 training sets. 146
8-7 Mutual information estimates with and without dimension compensation. 148
8-8. Classification performance of QMI ranked features with forward selection for SON10 training set. 151
8-9. Classification performance of QMI ranked features with forward selection for SON10 test set. 151

8-10. Classification performance of QMI ranked features with backward elimination for SON10 training set. 152

8-11. Classification performance of QMI ranked features with backward elimination for SON10 test set. 152

8-12. Classification performance of QMI ranked features with forward selection for SON512 training set. 153

8-13. Classification performance of QMI ranked features with forward selection for SON512 test set. 153

8-14 Maximum QMI discriminative feature extraction....155

8-15 Maximum Linear QMI feature extraction for SON10 training set. 157

8-16 Maximum Linear QMI feature extraction for SON10 test set. 157

8-17 Maximum Linear QMI feature extraction for SON512 training set. 158

8-18 Maximum Linear QMI feature extraction for SON512 test set. 158

8-19 Class feature space in 2-D for SON10 prior to training. 159

8-20 Class feature space in 2-D for SON10 after NLQMIC training. 160

8-21 Comparison of 1-D PDF for SON10 maximum QMI feature extraction. 161

8-22 Comparison of 1-D PDF for SON512 maximum QMI feature extraction. 161

8-23 Minimum error bound comparison of LDA and QMI feature extraction. 163

8-24 Comparison of CSS algorithms and QMI feature extraction for SON10....164

8-25 Comparison of CSS algorithms and QMI feature extraction for SON512....164

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

AN INFORMATION-THEORETIC APPROACH TO SONAR AUTOMATIC TARGET RECOGNITION

By

Rodney Alberto Morejon

May 2003

Chair: Dr. Jose C. Principe
Major Department: Electrical and Computer Engineering

Multistage automatic target recognition (ATR) algorithms provide an effective means for locating items of interest in sonar imagery by successively reducing the volume of data at each stage in order to produce a concise list of possible target locations. Traditionally, the stages in these algorithms are designed using statistical methods that are often based on a priori assumptions about the input data distributions. By extending recent advances in information-theoretic learning (ITL), this dissertation applies concepts of information theory to the feature extraction stage of the multistage sonar ATR paradigm. By incorporating information-theoretic performance metrics, the approach is not dependent on a priori statistical assumptions about the data distributions.

The widespread application of ITL has likely been hindered by two phenomena: computational complexity and difficulty of use. This dissertation addresses both of these issues in order to increase the applicability of ITL. First, the computation of the ITL criteria is optimized for more efficient evaluation. Next, training efficiency is improved

by tailoring some of the most popular advanced training algorithms for compatibility with ITL systems. To further improve training efficiency, a batch training approach to ITL is presented that reduces the quadratic complexity of the criterion. Finally, various properties of the information-theoretic criteria are investigated in order to achieve a better understanding of the ITL process in support of easier implementation.

These advances in ITL efficiency and understanding are then applied to address the problem of sonar feature extraction for ATR. This problem represents a difficult real-world challenge for the application of ITL and is representative of the class of problems that stand to benefit from a more widespread implementation of ITL principles. The information-theoretic feature extraction methods presented are shown to be more effective and robust than the most popular statistical methods and some of the currently fielded sonar ATR algorithms. The ITL methods presented provide an alternative approach for designing the subspace mapping functions that are prevalent throughout ATR systems.

CHAPTER 1
INTRODUCTION

The use of automatic target recognition (ATR) algorithms for the detection of items of interest in sonar data is a formidable problem that has received much attention from the defense, medical, and academic communities. Typically, a multistage approach is implemented that systematically reduces the volume of data from stage to stage by applying increasingly complex algorithms at each stage. Many of the traditional methods for designing these algorithms have been based on statistical methods that make a priori assumptions about the input data distributions. Since the information-theoretic learning (ITL) paradigm was developed specifically to avoid these assumptions [Pri00], an information-theoretic approach to sonar ATR promises the potential for more robust detection performance.

In the preliminary phases of this research, the initial goal was to develop novel approaches (based on information-theoretic criteria) for addressing the functionality of several stages in the multistage sonar ATR paradigm. Originally, the motivation was to highlight the versatility of ITL across a wide range of mapping functions in the multistage ATR process. However, initial experimentation with real sonar datasets highlighted some of the practical challenges with applying ITL techniques to larger scale problems. The most pronounced challenge was the computational complexity: many minutes were needed to compute a single information-theoretic metric. The lack of implementation guidelines for selecting the parameters associated with ITL also

contributed to the original difficulty of use. These challenges helped to reshape the focus of this research accordingly. The ultimate objectives evolved into the following:

- Reducing the computational complexity of the ITL process.
- Increasing the understanding of ITL in order to facilitate the implementation process.
- Applying ITL to one stage of the sonar ATR paradigm, namely, the feature extraction stage.

The remainder of this chapter presents some basic background information on sonar ATR and ITL along with the motivation, from both application and academic perspectives for the work presented herein. An outline of the remaining chapters of the dissertation concludes this chapter.

Background

The problem of identifying items of interest in sonar images is of prime concern to many groups ranging from the medical professional to the military. Specially trained human operators are called upon to perform the tedious task of examining sonar images to locate and identify targets or items of interest. In certain applications, particularly in defense related areas, the need for continuous monitoring of a real-time data stream makes this is a very demanding and stressful task requiring frequent shift changes in order to maintain fresh eyes. In order to alleviate the stress on human operators, ATR algorithms have been developed to assist, supplement, and in some cases completely replace, the man-in-the-loop. These algorithms do not get tired and always maintain a constant vigil.

The overall utility of an ATR algorithm is typically measured using metrics of target detection performance, run-time computational efficiency, and training efficiency. The detection performance measures how well the algorithm can separate items of

interest from the background or clutter. Detection performance is usually characterized as a function of detection probability and false alarm rate using a receiver operating characteristics (ROC) curve. In many applications, sonar target analysis requires that an ATR algorithm rapidly or continuously process large volumes of data. In these cases, the algorithm's utility is measured not only by detection performance, but also by its ability to complete its processing as quickly, or efficiently, as possible so as to keep up with the input data stream. For these cases, computational efficiency is also of great importance in ATR design. Lastly, the training efficiency in terms of the volume of input data and the time required for training can also be limiting factors as to the utility of a given ATR design.

Many sonar ATR algorithms have been designed using a variety of theoretical methods including detection theory, statistical pattern recognition, matched filtering, neural networks, wavelet transforms, and model-based methods [Ari98, Dob97, Sac99, Syz99]. In general, most of these algorithms take a multistaged approach consisting of the following or similar stages: a preprocessing or image normalization stage, a coarse detection or clutter rejection stage, a feature extraction stage, a classification stage, and a fusion stage if multiple sensor sources or algorithms are available. From one perspective, each of these stages can be viewed as an information filter or mapping function that seeks to filter out the information that is not relevant to the next stage of ATR, thus reducing the dimension and/or volume of the data at each successive stage.

Interestingly, little work has been done that applies the concepts of information theory to the design of ATR systems. Historically, this was mostly because of the increased computational complexity that is a consequence of most information-theoretic

implementations. However, recent developments in information-theoretic learning systems have led to a more computationally tractable approach to applying information-theoretic criteria to trainable systems [Pri00]. While these developments have significantly advanced the feasibility of ITL systems, two difficulties still hamper more widespread applicability of ITL systems. First, the computational complexity of the criterion function, although significantly reduced from previous implementations, is proportional to the square of the number of samples. While this complexity is acceptable for applications with small datasets, it can quickly become computationally prohibitive with some of the larger datasets that are commonly encountered in real world applications. The second problem most likely stems from the relative newness of the ITL method. Although the method has been applied to solve a wide range of problem classes, these solutions have been developed almost exclusively by experts in the field. Little work has been published to establish general guidelines for ITL implementations or to provide a better understanding of the information estimators so that the concepts of ITL can be used by a more mainstream group of users.

The primary objective of this dissertation is to advance the science of ITL further by improving the computational efficiency and understanding of the ITL process. These advancements in efficiency and understanding serve as tools that allow information-theoretic concepts to be applied to the sonar ATR problem. Within the context of sonar ATR, the efforts of this research focus on the feature extraction stage with the objective of developing novel methods for feature extraction that provide improved detection performance while optimizing the computational and training efficiencies of the ITL

process. The utility of these methods is measured through comparisons with some existing and common approaches to the feature extraction problem.

Motivation

Existing Navy sonar ATR algorithms provide detection performance on a level that is comparable to that of a human operator [Dob97]. In many cases, the computational requirements of these algorithms are very high. The Navy has identified the need for improving the performance of existing ATR algorithms. It has funded various research and development efforts at the Coastal Systems Station to investigate approaches for improving the detection performance and efficiency of ATR algorithms. Improving detection performance brings significant and obvious benefits to the Naval fleet (such as increased safety, reduced losses, and faster fleet response times). Improved computational efficiency can allow for real-time vs. offline data processing, can reduce cost and complexity, and can provide for feasible simulation-based analysis. Better training efficiency helps to speed up the design process and can potentially allow for the use of more complex detection algorithms. By working closely with recognized experts in Naval sonar ATR and ITL, the results of this dissertation can potentially provide advancements to the field of sonar ATR in these areas.

Many of the functions that take place within the various stages of an ATR system share a common goal – to create a mapping from the input space to a synthetic space that optimizes some figure of merit:

- Normalization creates a full rank projection that produces some desired background statistics.
- Feature extraction creates a subspace projection that possesses informative or discriminative characteristics of the input data.

- Classification creates a subspace that optimizes some measure of class membership.

These similarities motivate the investigation of a general approach to address this common functionality. Further motivation is provided by the fact that, in many cases, the traditional approaches used to address these functions are based on statistical assumptions that are not valid in general.

Information theory provides an attractive alternative approach to these issues. First, ITL methods make no assumptions about the probability density functions of the data. This eliminates the errors associated with incorrect model assumptions. Second, ITL criteria are very flexible and can be used in various supervised and unsupervised ways. This flexibility makes the approach readily applicable to a wide array of mapping function problems to include feature extraction, data fusion, and classification. The last, and perhaps most attractive, feature of ITL is due to the very nature of information theory. Information theory concerns itself with the information content of messages, optimum information compression, and information equivalence. These concepts are more complex and robust for information processing than more simplistic measures such as mean-square error (MSE). Since each stage of an ATR system is effectively an information filter, designing them based on information-theoretic criteria is a natural conclusion.

For the purpose of maintaining focus and limiting the scope of work presented, the results in this dissertation are constrained to the feature extraction stage of ATR. However, the approaches and methods described herein can be readily applied and extended to other stages of the ATR problem. In fact, ongoing research is currently

investigating the use of ITL concepts for the data fusion and classification stages of the sonar ATR problem.

Overview of Dissertation

Chapter 2 includes an overview of the general multistage ATR approach, and describes some recent implementations that have been used specifically for sonar ATR with special emphasis placed on the feature extraction stage. Chapter 2 also includes an overview of the characteristics of sonar imagery along with a description of the sonar datasets that have been made available to support this work. Chapter 3 explains the basic concepts of Information Theory, the application of information-based criteria to general learning systems, and summarizes the ITL approach of Principe et al. Chapter 4 presents an optimized implementation of the entropy and mutual information criteria estimators for ITL. In Chapter 5, training efficiency is addressed by adapting several popular advanced parameter search techniques for use with ITL systems. Chapter 6 introduces a batch approach to training ITL systems that reduces computational complexity by partitioning the dataset into smaller batches. Chapter 7 examines some of the properties of the ITL criteria estimators, explores the relationships between the estimator parameters, and helps to increase an overall understanding of the ITL process. Chapter 8 presents several information-theoretic techniques to address the feature extraction problem using real world sonar data. The Chapter 9 summarizes the contributions of this research and presents avenues for future research.

CHAPTER 2
SONAR AUTOMATIC TARGET RECOGNITION (ATR)

Introduction

The development of automated techniques for target recognition has been the subject of much research for a wide range of sensor systems. In the context of underwater sonar imagery, private industry and the military have supported the development of many algorithms that attempt to reduce the need for or improve the performance of a man-in-the-loop. Throughout this body of research, the multistage paradigm has emerged as the virtually universal underlying architecture for most implementations [Ari98, Cha99, Dob97, Guo99, Hyl95, Lau98]. This chapter presents a summary of the multistage sonar ATR process with special emphasis on the feature extraction stage. First, an overview of the general characteristics of sonar imagery is presented. Next, the typical multistage ATR process is described for general sensor systems. This is followed by a description of the common methods that have been developed to address the various stages of ATR for sonar imagery. The next section focuses more closely on the feature extraction problem and reviews some of the most common methods and terminology. The chapter concludes with a description of the real-world sonar datasets that have been made available to support this research.

Underwater Sonar Image Characterization

Successful target recognition in two-dimensional imagery is highly dependent on the characteristics of the type of sensor used to produce the image in addition to the environmental conditions encountered during collection. Sonar imagery has unique

characteristics that distinguish it from data of other sources such as radar, infrared, and visible light. Underwater sonar images are collected from a vehicle moving through the water with a sonar transceiver. The transmitter emits a periodic signal of acoustic energy, or ping, while the receiver listens for the return signal. The image is formed sequentially by row, or ping-by-ping, as the vehicle makes headway through the water. The following features typically characterize most sonar imagery:

1. Attenuation in signal with increasing range.
2. Banding caused by automatic gain correction.
3. Motion effects caused by nonlinear vehicle movement.

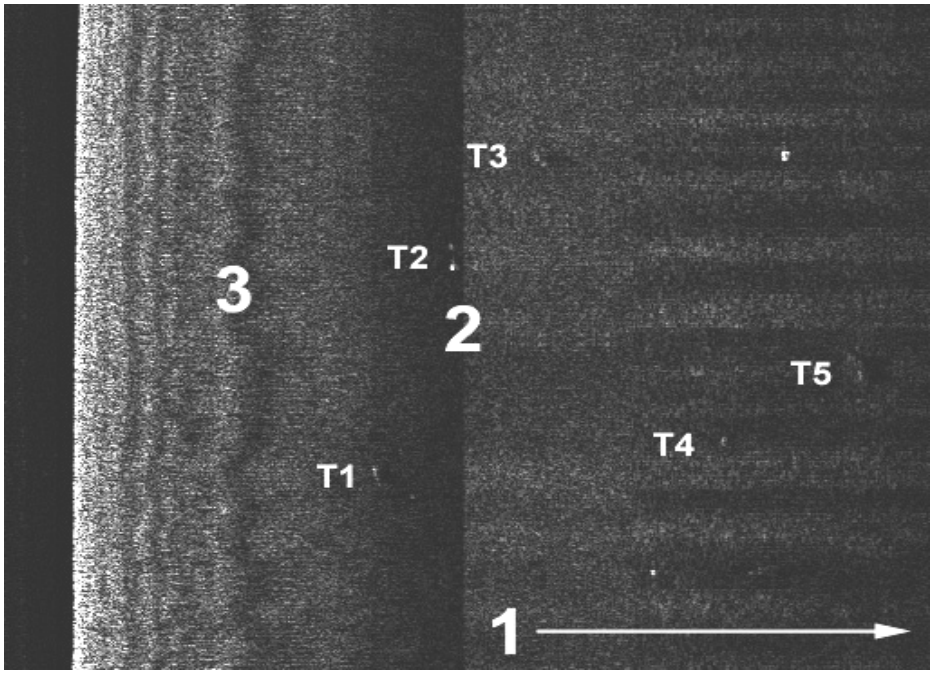

Figure 2-1. Representative sonar image.

Figure 2-1 shows a representative starboard (right) side-scan sonar image with the vehicle track moving up the left side of the image. The figure shows the three general features identified above marked with their respective numbers. In sonar ATR, the objective is to locate the objects of interest, or targets, while rejecting the background, or acoustic clutter. Environmental conditions that contribute to acoustic clutter include

physical bottom clutter, reverberation, and multipath returns [Nel96]. Targets in sonar imagery are typically identified by a bright highlight followed by a dark shadow. Figure 2-1 contains five such targets designated by the $\mathbf{T}_n$ markings. The targets are immediately to the right of the labels. As is evident from the figure, sonar images typically contain nontarget highlights and shadows from acoustic clutter that often result in false alarms, or false target indications, when the image is processed.

General Multistage ATR

One of the most prevalent solutions to the ATR problem is the multistage approach. This approach uses multiple algorithms that attempt to reduce the data volume from stage to stage. A typical example of this approach is shown in Figure 2-2. Generally, the complexity of the algorithm increases from left to right as the data volume decreases. For example, the detector will usually process the entire image with a simple algorithm while the classifier will focus a more complex algorithm on a small fraction of the image or on a set of features extracted from the image. The five most common components include: the preprocessor, the detector, the feature extractor, the classifier, and the fusion stage. Brief descriptions of the general functionality of these stages are listed below.

Preprocessor. The preprocessor stage of an ATR is typically used to apply some type of normalization technique over the entire image. Even though normalization is normally not a highly complex task, the fact that it is usually applied to every pixel can make it a significant contributor to the overall computation total.

Detector. The detector, sometimes referred to as a prescreener or focus of attention stage, is typically responsible for the greatest reduction of data volume in the ATR process. Since it is applied over the entire image, algorithm efficiency is critical in

keeping the total computations in check. The detector is usually set to a very high PD (90-100%) in order to allow most of the targets through to the more sophisticated stages of processing.

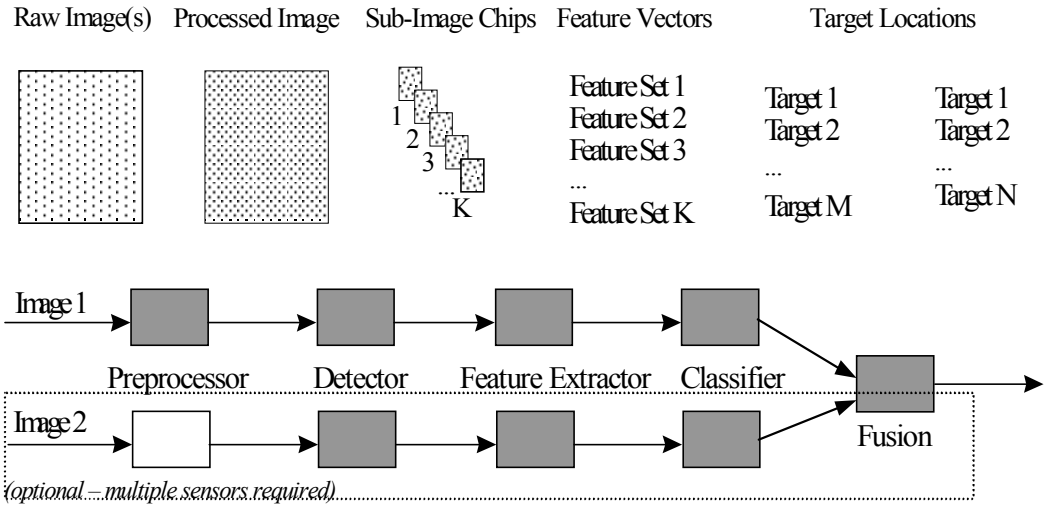

Figure 2-2. Multistaged ATR approach.

Feature extractor. In the feature extractor, data reduction is achieved through information abstraction or compression rather than by data screening (as in the detector). This stage is used to convert the pixel representation of the screened subimages into a more concise feature set that represents the salient characteristics of the subimage. Some of the features might relate to physical characteristics of the target such as size and orientation. Other features might represent statistical characteristics of the subimage.

Classifier. The classifier stage is typically the final stage of an ATR. With the list of feature vectors as input, it is responsible for determining which feature sets correspond to targets and which represent false alarms. Classifiers can be as simple as a threshold operation and as complicated as nonlinear decision surfaces. The greater the number of features in the feature vector, the more complex a classifier design can be.

Fusion stage. In cases where more than one image, sensor, or algorithm is present, a fusion stage is often used to combine the results. Typically, the results are combined after the classifier using some statistical means, particularly if the classifier can provide confidence levels for each target nomination. Other fusion techniques use logical operations such as the AND function or fuzzy logic techniques. Although fusion is described here as a post-classification stage, in practice it can be applied at any stage of the ATR process as will be discussed in the sections below.

Sonar Imagery ATR Methods

Preprocessing/Image Normalization Approaches

Upon inspection of Figure 2-1, it is reasonable to conclude that image normalization would help compensate for the range attenuation and bring the darker targets out of the shadows. In general, this is the first stage of most ATR approaches reviewed in the literature [Ari98, Cha99, Dob97, Guo99]. Most normalization techniques attempt to bring the background to a constant level throughout the image so that shadow and highlight levels are compared to a consistent background in subsequent stages of ATR. The results of a sequential ping-by-ping approach shown in Figure 2-3 are typical. Clearly, the targets are more visible to the human eye in the normalized image. Other normalization techniques applied to sonar image analysis include: range based normalization, forward and backward filtering [Dob99], logarithms [Lan00], and low pass median filtering [Dob97].

In addition to normalization, other techniques are sometimes used as part of the preprocessing stage of sonar ATR. The most common of these is image decimation through pixel combinations [Ari97, Jes99, Mor97]. This step sacrifices image resolution for noise reduction and computational complexity reduction. The most common

decimation technique is pixel addition, however, Aridgides uses the 'largest of,' or ORing function, for the pixels in each decimation block. The effect of these decimation approaches is similar to the wavelet transform methods that are discussed below.

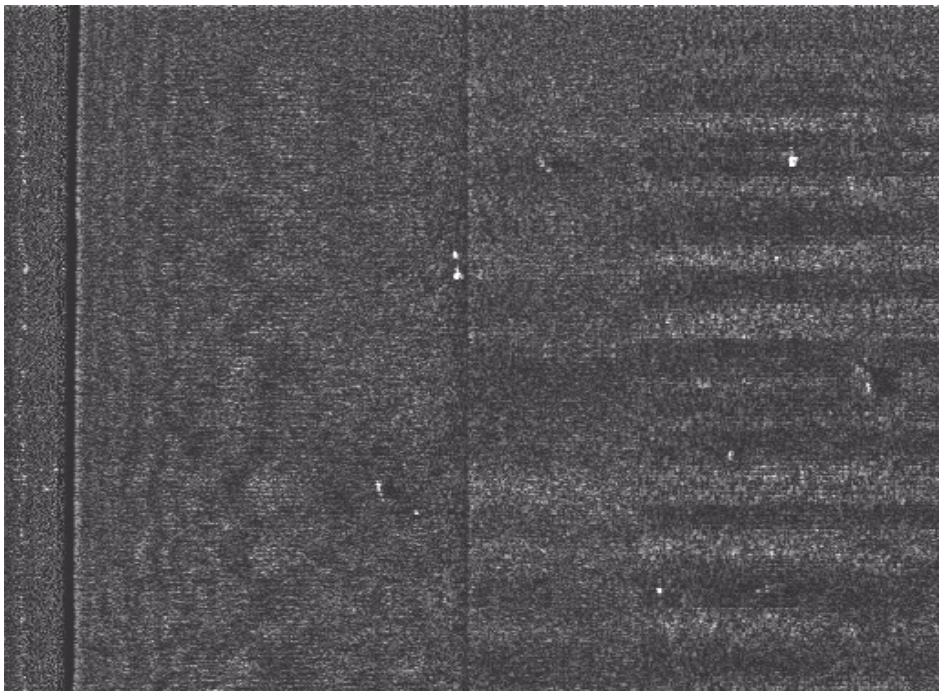

Figure 2-3. Normalized sonar image.

Detection Approaches

The detection stage usually results in the greatest reduction in data volume. Since it is applied to the entire image, it is desirable to keep the computational complexity of this stage to a minimum. The detector typically applies a simple statistical threshold operation to a mask that scans the image. Since the threshold operation may be highly dependent on any preprocessing or normalization, detection is often closely tied to the preprocessing stage. Varieties of approaches have been applied to the detection stage of sonar ATR. Common algorithms combine highlight and shadow detection schemes that look for pixel counts and intensities that are a threshold above or below the levels of the normalized background. Dobeck et al. applied a range-dependent nonlinear matched

filter as part of the Advanced Mine Detection and Classification (AMDAC) algorithm that attempts to match pretarget, highlight, dead zone, and shadow areas across the image [Dob97]. Bello proposed a random Markov field anomaly detector [Bel95b]. Lau and Chao combined a morphological filter and simplified Holmes double-gated filter with image sieves (based on wavelet transforms) that pass subimages of the appropriate size. The Holmes double-gated filter, also referred to as the two-parameter constant false alarm rate (CFAR) detector, is commonly used as a detection algorithm in ATR systems for its simplicity and performance in variably cluttered imagery. The typical CFAR stencil includes a center window for the area under test along with a surrounding band to compute local statistics. The detector uses a simple threshold to accept candidate target areas based on (2-1) that essentially compares the average intensity of the area under test with the average intensity of the local area normalized by the standard deviation of the local area.

$$\frac{\overline{X}_{test} - \overline{X}_{local}}{\sigma_{local}} > threshold \qquad (2\text{-}1)$$

Feature Extraction Approaches

Typically, in an ATR system, the Detector and Classifier achieve data reduction by eliminating candidate target areas from further contention as possible targets. The feature extractor also performs data reduction, but in a different way. The feature extractor converts an image or subimage from a pixel representation to a feature list that is designed to represent the key characteristics, or features, of a potential target. Like detection algorithms, feature extraction algorithms are often tied to the previous stages of their particular ATR processing string. It is common, for example, to use pixel statistics or coefficients that were determined as a consequence of the preprocessing or detection

stages as part of the feature set that is passed on to the classification stage. Common features of this type include highlight and shadow pixel counts, intensities from both raw and filtered images, and wavelet coefficients. Other features, describing the physical characteristics of the detected object, are often extracted from the raw or filtered image. These features include items such as object length, width, and orientation. Additionally, features are often extracted based on some statistical or mathematical properties of the detected objects. For example, Lau and Chao extract the Hermite coefficients from each detected subimage as their primary feature set [Lau98]. Kronquist et al. compute a large base of wavelet coefficients and select the most discriminant basis [Kro99]. Chandran combines pixel statistics with higher order spectral features including bispectrum and trispectrum [Cha99].

In many cases, all the extracted features are passed on to the classification stage. However, in some instances, feature reduction is performed to reduce the dimensionality of the feature set and thus avoid Bellman's curse of dimensionality [Bel61]. This is an important issue in sonar ATR at this point since the available datasets are limited in size. Using too many features leads to poor generalization while too few features may not convey enough information to discriminate between classes. Dobeck et al. explore selecting the five most discriminating features incrementally in addition to a combined forward and backward search [Dob97, Hyl95]. Chandran combines his feature set using a variation of principle component analysis with three classes [Cha99]. Aridgides uses a feature orthogonalization procedure to derive independent features from the original set [Ari98]. The next major section describes the feature mapping process in detail and describes some of the most common approaches to feature extraction and selection.

Classification Approaches

In general, the classifier stage is designed to take the selected feature space and partition it into target and nontarget subspaces based on the available training vectors. By far, the most common implementation for the classifier is the multilayer perceptron (MLP) neural network trained with MSE optimization criteria and backpropagation learning rule [Guo99, Lau98, Sac99, Szy99]. One novel approach is Dobeck's k-nearest neighbor (KNN) neural network containing attractor and confidence layers [Hyl95]. This network uses the training set to determine the location, number, and size of the attractors, which are based on radial basis functions. The confidence layer computes the likelihood that a feature set belongs to a given class. The KNN classifier is quickly trained by noniterative methods and is computationally efficient during run-time as well. Dobeck subsequently combined the KNN classifier with an optimal discriminatory filter classifier (ODFC) base on linear discrimination theory by ANDing the two outputs [Dob97]. Chandran proposed a sequential combination of thresholding, minimum distance, and KNN classifiers. Guo proposed a wavelet channel clustered MLP topology that limits the full connectivity of certain neurons to specific wavelet coefficients [Guo98]. Aridgides uses an implementation of the log likelihood ratio test combined with the precomputed histograms of orthogonalized features derived from the training set.

Fusion Approaches

Fusion can take place at various stages and in several ways within an ATR system. The two major categories include multisensor fusion and multiple algorithm fusion. Multisensor fusion combines different data sources (which can be processed with the same or different algorithms), while multiple algorithm fusion combines the application of different algorithms to the same data. Combination of low frequency and

high frequency sonar imagery is multisensor fusion, while Dobeck's approach of ANDing two classifiers that are trained with the same data is an example of algorithm fusion. Multisensor fusion can take place at any stage in the ATR from preprocessing to postclassification. Aridgides proposed a predetection 3-d adaptive clutter filter that combines high and low frequency sonar imagery for subsequent detection stages [Ari97]. Although this approach realized improvements over individual sensor ATR, he subsequently realized better results with a postclassification fusion using simple ANDing [Ari98]. Dobeck used a postclassification fuzzy ANDing based on class confidence measures that also produced significant improvements over individual sensors. Sacramone uses a postclassifier neural network to combine multiple sensors that looks at the combined feature space for both sensors [Sac99]. The benefits of algorithm fusion in reducing false alarm rates were demonstrated separately by Dobeck, Huang, and Aridgides when they combined full ATR processing strings from Coastal Systems Station (CSS), Raytheon, and Lockheed Martin [Ari00, Dob00, Hua00]. Gou and Symczak also demonstrated improved performance by combining multiple ATR processing strings on the same input data [Guo99].

Feature Extraction Overview

The problem of extracting features from datasets, also referred to as feature mapping, is prevalent across a wide range of fields from control systems to signal processing to pattern recognition. The goal is to find a mapping function, $g : \mathcal{R}^K \rightarrow \mathcal{R}^M$, from one space to another described by the equation, $\mathbf{Y} = g(\mathbf{X})$, where $\mathbf{X} \in \mathcal{R}^K$ and $\mathbf{Y} \in \mathcal{R}^M$ that optimizes a given criterion. The mapping function can be linear (e.g., matrix) or nonlinear (e.g., MLP). The mapping can be volume preserving when $M=K$

(e.g., digital filtering and normalization), volume reducing when $M<K$ (e.g., feature extraction, feature selection, and compression), or volume expanding when $M>K$ (e.g., decompression and reconstruction). The goal of the mapping criteria can also take on a variety of forms including signal representation, class discrimination, and information fusion. For this discussion, the primary focus will be in the volume reducing and preserving mappings for feature extraction.

The feature extraction process can be either data-driven or based on prior knowledge or use some combination of these methods. Data driven feature extraction relies exclusively on the data, the form of the mapping function, and the mapping criteria to arrive at the set of new features. In other cases, when relationships are known to be important from prior knowledge, it is often useful to generate features based on this information. For example, the ratio between height and width of an object might be a significant factor in discriminating between object classes. It is often much more efficient to compute these features directly than to depend on the mapping process to extract some representation of this phenomenon as in the data-driven method.

Feature extraction for dimension reduction can be done by discarding a subset of original data, known as feature selection, or through linear or nonlinear combinations of original data. One example in image processing is pixel averaging. In most situations, dimension reduction results in loss of information. Therefore, one of the main goals in designing a good feature extraction system is to ensure that as much of the relevant information as possible is retained [Bis95]. In feature extraction for pattern recognition, particularly when the input is a high dimensional space relative to the amount of training data, one of the primary goals is to reduce the effects of the curse of dimensionality

[Bel61]. As described by Bellman, this curse refers to the exponential growth in the need for training data as the dimension of the feature space increases. In the typical case where limited training data exists, this is addressed with a volume reducing mapping function where the minimum number of features, M, is sought that can still adequately represent the information to the subsequent process. If the subsequent process is to design a classifier that is expected to generalize to unseen data sets, care must be taken not to impose class separability as the mapping criteria until the dimensionality of the features has been adequately reduced. Failure to do so often results in poor generalization performance.

Fukunaka distinguishes between feature mappings for signal representation and mappings for classification [Fuk90]. Rubinstein and Hastie refer to these as informative and discriminative classification systems, respectively [Rub97]. In the context of feature selection, Koller and Sahami refer to these as filter and wrapper methods, respectively [Kol96]. An informative, filter, or signal representation mapping, does not take into account class information during feature extraction and is concerned with finding the principle descriptive components or class feature densities of the data. Discriminative mappings focus on finding the class boundaries with no consideration for signal representation. The sequential combination of informative and then discriminative mappings can be used to capture the strength of both methods. However, care must be taken so as not to remove the discriminative features of the data during the volume reduction of the informative stage. In practice, this is difficult to ascertain. The sections that follow describe several of the most commonly used methods for both informative and discriminative feature extraction.

Feature Selection

One of the most basic implementations of feature extraction is feature selection. Feature selection is a straightforward linear volume reducing method that simply selects a subset of the original features as the new feature set with the goal that they best represent or discriminate the original signals. Selection of the best M out of N features is a combinatorial problem that quickly becomes prohibitive with increasing dimension. Common search techniques for avoiding exhaustive methods for feature selection include forward selection and backward elimination. Forward selection selects the most relevant feature from those remaining to add to the new feature space. Conversely, backward elimination drops the least relevant feature from further consideration. However, neither method guarantees the optimal set. Branch and bound methods are guaranteed optimal for monotonic criterion. Independent features allow for the selection of the M individual best features.

Feature selection is useful if some inputs carry little relevant information or if there are strong correlations between sets of inputs so that the data is redundant. The feature selection criterion for classification problems is optimally the probability of misclassification or total risk, but in practice, this may be too difficult to compute or estimate. Simplified criteria include measures of class separability based on covariance matrices. Some of these criteria are monotonic with respect to dimension, so they are not useful for comparing between different dimensionalities of feature sets. Cross validation techniques can be used in these cases [Bis95].

Koller and Sahami proposed an information-based approach to filter selection where the goal is to minimize information loss between the full feature set and the subset. The information loss is estimated by the cross-entropy between individual features.

Features are rejected using a backward elimination strategy that rejects the feature with minimum cross-entropy with respect to a predetermine number of features [Kol96].

Principal Component Analysis (PCA)

Principal component analysis, also called the Karhunen-Loeve Transform, is a linear, informative, unsupervised method for creating a feature mapping for signal representation. Typically, when used for feature extraction, the PCA mapping reduces the dimensionality of the feature space. The objective is to represent the input signal as closely as possible with a linear combination of a reduced set of orthonormal vector as follows,

$$\mathbf{X} = \sum_{i=1}^{N} y_i \mathbf{u}_i \cong \sum_{i=1}^{M} y_i \mathbf{u}_i + \sum_{M+1}^{N} b_i \mathbf{u}_i \qquad (2\text{-}2)$$

where the $\mathbf{u}_i$ form an orthonormal basis to represent $\mathbf{X}$, the y_i are the feature values, and the b_i are constants. In the approximation, the first M basis vectors, or principal components, are selected that minimize the sum of the square of the error in signal representation. It turns out that these principal components are the eigenvectors that correspond to the M largest eigenvalues of the covariance matrix of $\mathbf{X}$. An interesting property of PCA is that the effectiveness of each feature for signal representation is determined by its corresponding eigenvalue. In addition, the output features are uncorrelated and in the case where $\mathbf{X}$ is normally distributed, the features are independent [Fuk90].

Also of note is that PCA is the solution that arises when the goal is to either: minimize information loss, maximize output entropy, or maximize mutual information between the input and outputs, when the distribution of $\mathbf{X}$ is normal [Dec96].

Independent Component Analysis (ICA)

Independent component analysis is a generalization of PCA that seeks to extract features that are as independent as possible. The problem is to find a mapping, g(**X**), that maximizes some measure of the statistical independence of the output features. Deco and Obradovic propose two criteria for evaluating statistical independence: low-order cumulant expansion and mutual information between the output components. The cumulant method is based on satisfying statistical independence properties in the Fourier space. Under Gaussian assumptions, the mutual information approach reduces to minimization of the sum of the output variances. Without Gaussian assumptions, an Edgeworth expansion is proposed for estimation of the output entropies. It should be noted, however, that both methods are based on the restrictive assumption that g is an invertible volume preserving mapping [Dec96]. Bell and Sejnowski propose a nonlinear approach to ICA for source separation problems. They develop a learning rule for adapting neural network weights that is based on maximizing the output entropy [Bel95a].

Linear Discriminant Analysis (LDA)

Linear discriminant analysis is a linear, discriminative method for performing a dimension-reducing feature mapping that attempts to preserve as much class discriminatory information as possible. The goal is to find the optimum linear transformation matrix, **A**, that produces a new M-dimensional space, Y, according to the relationship,

$$\mathbf{Y}_{Mx1} = \mathbf{A}^{T}_{KxM}\mathbf{X}_{Kx1} \tag{2-3}$$

where M+1 is the number of classes. For the case of a two-class problem, the

transformation matrix is simply a vector. The method looks for the linear transformation that minimizes the within-class scatter, $\mathbf{S}_w$, while maximizing the between-class scatter, $\mathbf{S}_b$. It turns out that the LDA features are the eigenvectors that correspond to the eigenvalues of the product of the inverse of $\mathbf{S}_w$ with $\mathbf{S}_b$ [Fuk90].

Other Common Methods

One common approach for feature extraction is to make use of attributes derived form applying basic statistical or mathematical functions to the data. Some of these features include means, medians, variances, moments and other higher order statistics. In some cases, histogram bin values, ratios, or geometries are used, particularly where prior knowledge dictates the importance of the feature. Another approach is to use the coefficients of an alternate or transformed version of the input. For example, features such as wavelet transform, DCT, or FFT coefficients are commonplace. These transformations of the input data are often desirable because of the inherent properties of the transformation space. In practice, these features are often readily available because they have already been computed to support other processing stages.

Sonar Datasets

Sonar Set 10 (SON10)

The SON10 data set contains a mix of 404 starboard side and port side images collected using a typical side-scan sonar system. Each image contains 712 range cells and 1000 cross-range cells. Images are 8-bit (0-255) gray scale. The images contain 69 targets; some images have none, some have more than one. Target-like nontargets appear throughout the images. A sample image is shown in Figure 2-4.

This dataset is available in its raw format, as described above, or in a preprocessed stage where the normalization, detection, and baseline feature extraction

stages from the CSS-Navy ATR algorithm have been executed to generate a list of potential targets and feature sets. The preprocessed format provides a baseline for a fair comparison of some of the downstream stages of the multistage ATR process, such as the feature selection and classification stages. In this preprocessed format, SON10 contains 67 targets in 1047 exemplars in the combined training and test data files. Table 2-1 shows how these exemplars are distributed.

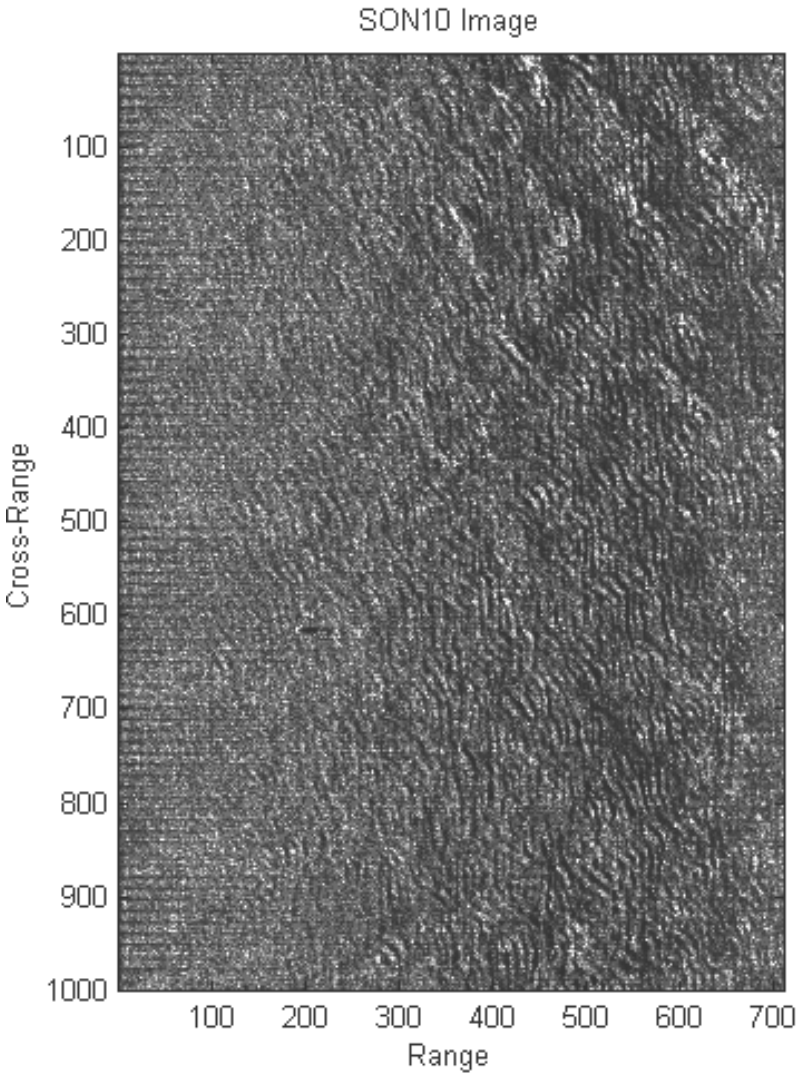

Figure 2-4. Typical SON10 sonar image.

Table 2-1. Distribution of extracted feature exemplars for SON10.

Set	No. Targets	No. Nontargets	Both
Train	24	447	481
Test	43	523	566
Total	67	980	1047

Sonar Set 5 and 12 (SON512)

The SON512 dataset contains a mix of 150 port side and starboard side images collected using a typical side-scan sonar system. Each image contains 640 range cells and either 900 or 1024 cross-range cells. Images are 8-bit (0-255) gray scale. The images contain 94 targets; some images have none, some have more than one. Target-like nontargets appear throughout the images. The near-range and far-range edge artifacts in these images are typically ignored. A sample image is shown in Figure 2-5.

This dataset is available in its raw format as described above or in a preprocessed stage where the normalization, detection, and feature extraction stages from the CSS ATR algorithm have been executed to generate a list of potential targets and feature sets. The preprocessed format provides a baseline for fair comparisons of some of the downstream stages of the multistage ATR process, such as the feature selection and classification stages. In this preprocessed format, SON512 contains 93 targets in 1182 exemplars in the combined training and test data files. Table 2-2 shows how these exemplars are distributed.

Table 2-2. Distribution of extracted feature exemplars for SON512.

Set	No. Targets	No. Nontargets	Both
Train	49	489	538
Test	44	600	64
Total	93	1089	1182

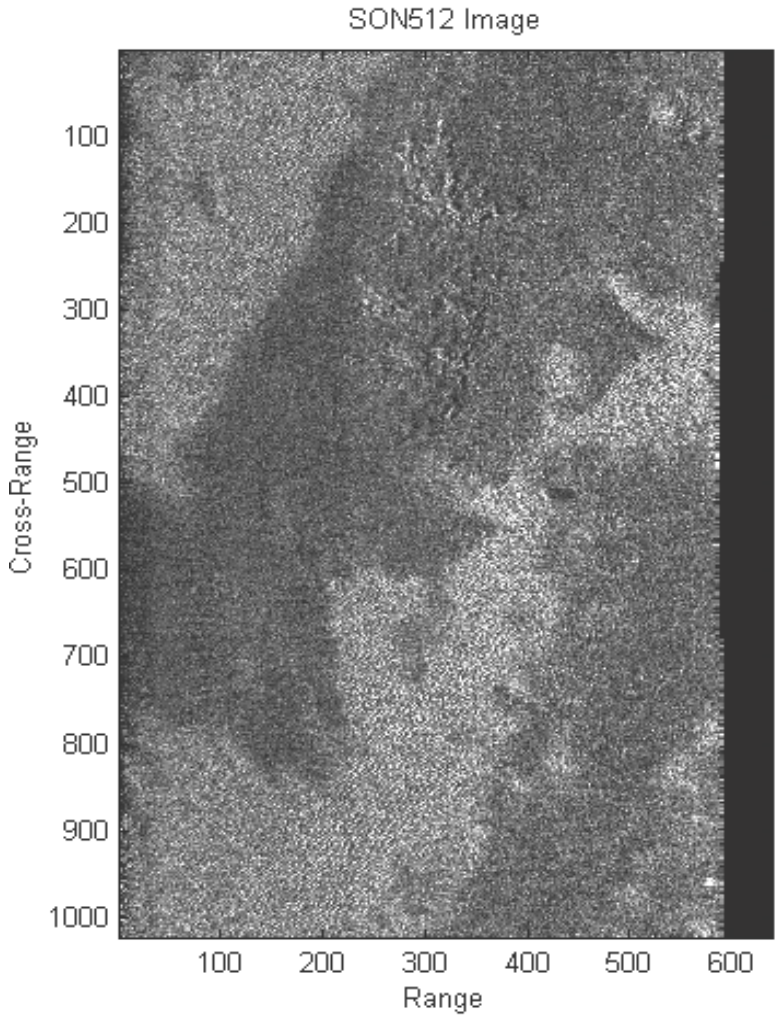

Figure 2-5. Typical SON512 sonar image.

Sonar Baseline Extracted Feature Sets

For both the SON10 and SON512 data sets, the CSS ATR algorithm has been used to establish a baseline feature set that has been made available to the research community for experimentation in the feature mapping, selection and classification stages. This feature set combines some of the statistical measures and count metrics extracted during the detection stage with some features specifically designed to extract important information based on prior knowledge. Table 2-3 summarizes the set of sixty-one features. Feature 15 should be ignored.

The baseline extracted feature sets for SON10 and SON512, as described in Table 2-3, serve as the input application data for this research. After the ITL approach is investigated and expanded throughout the next several chapters, Chapter 8 presents the application of ITL concepts to the feature extraction problem using the data sets described above.

Table 2-3. The CSS Baseline extracted feature set.

No.	Name	Description
1	Npix_nor	Number of normalized image pixels in the window that exceed a Threshold
2	Npix_mf	Number of matched-filter pixels in the window that exceed a Threshold
3	Max_pix_nor	Maximum normalized image pixel intensity in the window
4	Max_pix_mf	Maximum matched-filter pixel intensity in the window
5	HiLite_str_nor	Average highlight strength computed from the normalized image
6	HiLite_str_mf	Average highlight strength computed from the matched-filter image
7	Max_eig_nor	Length of major axis of an ellipse fit to highlight region of the normalized image
8	Min_eig_nor	Width of minor axis of an ellipse fit to highlight region of the normalized image
9	Max_eig_mf	Length of major axis of an ellipse fit to bright region of the matched-filter image
10	Min_eig_mf	Width of minor axis of an ellipse fit to bright region of the matched-filter image
11	Shadow_len	Shadow length
12	Shadow_str	Shadow strength
13	Max_pix_clu_mf	Maximum matched-filter intensity over the pixels in the detection Cluster
14	Npix_clu	Number of pixels in the detection cluster
15	Nclusters	Number of detected clusters in the image (measure of clutter density)
16 – 25	Nnor(i)	Number of normalized pixels above threshold(i) in the window
26 – 35	Nmf(i)	Number of matched-filter pixels above threshold(i) in the window
36 – 45	Nnor_diff(i)	Number of normalized pixels above threshold(i) in the window minus the number of normalized pixels above threshold(i) in the region that locally surrounds the window
46 – 55	Nmf_diff(i)	Number of matched-filter pixels above threshold(i) in the window minus the number of matched-filter pixels above threshold(i) in the region that locally surrounds the window
56 – 58	Avg_nor(i)	Average pixel intensity of normalized image over I-th window
59 – 61	Avg_mf(i)	Average pixel intensity of matched-filter image over i-th window

CHAPTER 3
INFORMATION-THEORETIC LEARNING (ITL)

Introduction

The concept of optimization using performance criterion derived from information theory has been around since Jaynes introduced the MaxEnt principle in 1957 [Jay57]. Since then, several metrics and architectures have been developed for the training of adaptive systems based on information theoretic measures [Bel95a, Dec96, Kul68, Lin89]. This chapter presents an overview of the information-theoretic learning paradigm along with a detailed description of the ITL approach that serves as the basis of this research. First, a brief review of some of the key concepts of information theory is presented. This section is followed by a description of a general architecture for learning systems that describes how performance criteria in general can be applied to their training. Next, a discussion of some of the most commonly used information-theoretic criteria is presented along with their typical applications. The chapter concludes with a detailed description of the ITL approach developed by Principe, et al.

Information Theory

Information theory provides a mathematical approach for analysis of the information content of messages and the effects of communication channel characteristics on message transmission. Concepts from information theory provide the answers to two key questions in communication theory:

- What is the minimum representation for a random variable?
- What is the maximum data rate achievable for a transmission channel?

These two questions are answered by: the entropy of the random variable, and the maximum of the mutual information between the input and output of the channel, respectively [Cov91].

Entropy is a measure of the uncertainty, or information content, of a random variable. The larger the entropy, the greater uncertainty exists about the actual value of the random variable. Greater uncertainty about the outcome means that more information is conveyed by the actual value of the random variable. It follows that random variables with larger entropy require more symbols (e.g., bits) for their representation. Shannon defined the entropy of a discrete random variable, X, with probabilities, $p(x)$, to be:

$$H_S(X) = -\sum p(x) \log p(x) \tag{3-1}$$

and

$$H_S(X) = -\int_x p(x) \log p(x) dx \tag{3-2}$$

for the continuous case where $p(x)$ is the probability density function (PDF).

Renyi [Ren76] proposed a generalized definition of entropy that includes Shannon's entropy as a special case. For a discrete random variable the form is:

$$H_{R\alpha}(X) = \frac{1}{1-\alpha} \log \sum p(x)^{\alpha} \tag{3-3}$$

and for continuous random variables,

$$H_{R\alpha}(X) = \frac{1}{1-\alpha} \log \left(\int_{-\infty}^{\infty} p(x)^{\alpha} \, dx \right), \; \alpha \neq 1 \tag{3-4}$$

where the relationship between Shannon and Renyi's given by,

$$\lim_{\alpha \to 1} H_{R\alpha} = H_S \tag{3-5}$$

$$H_{R\alpha} \geq H_S \geq H_{R\beta} \qquad \text{if } 0 < \alpha < 1 \text{ and } \beta > 1 \tag{3-6}$$

Of particular interest to the information-theoretic learning approach by Principe et al. is the case where $\alpha = 2$, or Renyi's quadratic entropy.

Mutual information is a measure of the dependence between two random variables, *X* and *Y*, and is defined by Shannon for the discrete case as:

$$I_S(X;Y) = H_S(X) - H_S(X \mid Y) = \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \tag{3-7}$$

and,

$$I_S(X;Y) = \int_x \int_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)} dydx \tag{3-8}$$

for the continuous case [Dec96]. This is equivalent to the Kullback-Leibler distance between the joint PDF and the product of the factorized PDF's. From (3-8), it can be seen that the mutual information is maximized when the conditional entropy between *X* and *Y* is minimized. This occurs as the factorized probabilities become similar, or $p(y) \rightarrow p(x)$. Thus maximizing mutual information, subject to the constraints of a particular communication channel, provides the maximum data rate achievable through that channel or channel capacity. Renyi's dependence measure takes the form described by (3-9) for the continuous case.

$$I_{R\alpha}(X;Y) = \frac{1}{\alpha - 1} \log \int_x \int_y \frac{p(x,y)^\alpha}{(p(x)p(y))^{\alpha-1}} dxdy \tag{3-9}$$

The fundamental objective of information-theoretic learning is to apply performance criteria based on the information-theoretic measures described above to the adaptation of learning systems.

General Learning System Architecture

Learning or trainable systems can be viewed as a mapping function, $g : \mathcal{R}^K \rightarrow \mathcal{R}^M$, from one space to another described by the equation, $\mathbf{Y} = g(\mathbf{X}, \mathbf{W})$, where $\mathbf{X} \in \mathcal{R}^K$ represents the system input, $\mathbf{Y} \in \mathcal{R}^M$ represents the system output, and $\mathbf{W}$ represents the set of trainable parameters. Common examples of learning systems include neural networks and adaptive filters. The goal in training such a system is to select the parameters of the mapping, $\mathbf{W}$, that optimize some performance criterion in either a supervised or unsupervised manner. The most common supervised learning criterion is minimization of the MSE between the output and some desired signal. Other examples include the Minowski-*R* error (or L_R norm) and relative or cross entropy. Unsupervised learning criteria include Hebbian learning, Oja's rule, and competitive learning [Her91]. Figure 3-1 shows a block diagram of a general learning system architecture.

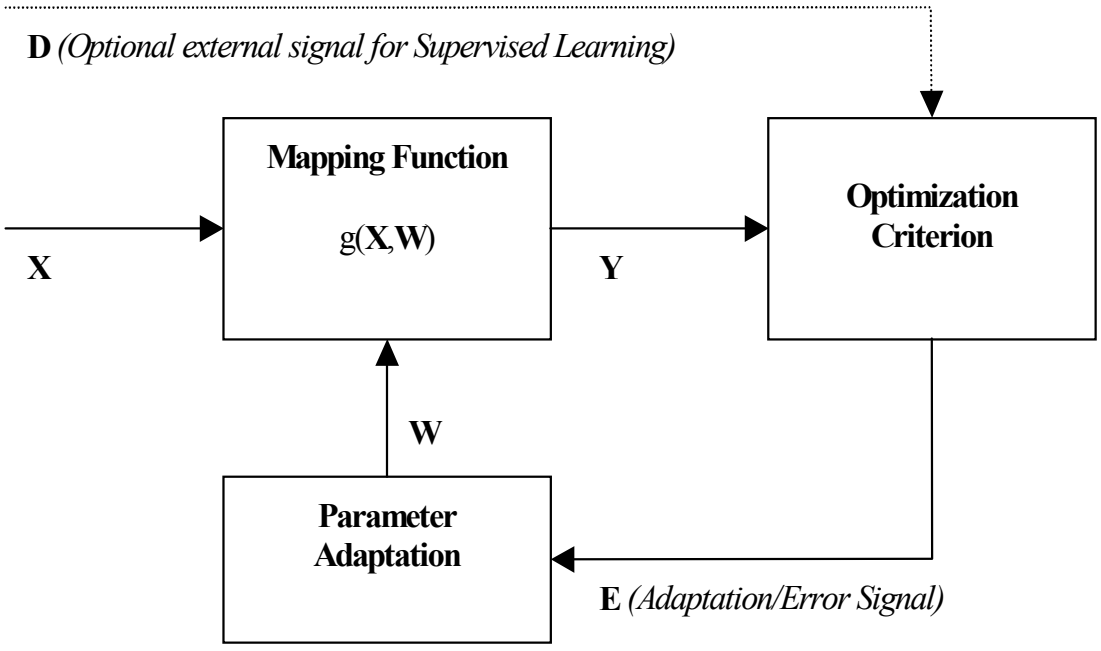

Figure 3-1. General learning system architecture.

One of the most prevalent implementations of Figure 3-1 uses MSE as the optimization criteria with a desired signal, a neural network as the mapping function, and back-propagation as the parameter adaptation algorithm. However, Figure 3-1 has many more applications than this. From the perspective of information theory, the architecture in Figure 3-1 can be viewed in a number of ways. First, the mapping function can represent an encoding function seeking to represent the input data in a more compact format. This is analogous to feature extraction or data compression. Then, in an unsupervised manner, the learning system can be trained to maximize the output entropy as its optimization criteria. This maximization process yields a representation of the input that conveys the maximum information content subject to the constraints of the mapping network. As a second example, the mapping function can be viewed as a communication channel through which we seek to transmit a signal. In this case, the learning system can be trained in a supervised manner to maximize the mutual information between the input and output signals. This produces the mapping parameters that yield the greatest channel capacity, again, subject to the constraints of the mapping network topology. Various information-theoretic criteria can be readily applied to the architecture in Figure 3-1. These are described in more detail in the following section.

Information Theoretic Optimization Criterion

Although the MSE criterion has well known computational advantages and is theoretically and practically appropriate for the solution of many problems, the assumptions inherent when using MSE can produce suboptimal results in many instances. The MSE criterion is derived from the principle of maximum likelihood by assuming that the distribution of the output data can be described by a Gaussian function with an input-dependent mean and a single global variance parameter. Thus, the MSE criterion cannot

distinguish between the true distribution, and a Gaussian distribution having the same input-dependent mean and average variance. This limitation leads to difficulties in inverse problems for multivalued target data and has been shown to be suboptimal for classification problems [Bis95].

These shortcomings motivate the use of alternative criterion for training of learning systems. By optimizing learning systems based on various information-based measures, the resulting information content is effectively optimized. The most common information based measures and their practical applications include:

- Maximization of Output Entropy (MaxEnt): MaxEnt is an unsupervised method that seeks to spread the output signal throughout the available output space, thus lending itself to dimensionality reduction, feature extraction, data compression, and/or preprocessing of input signal.

- Minimization of Cross Entropy or Mutual Information between Output Signals (MinXEnt): MinXEnt tries to reduce the dependence between signals seeking to make each output independent of one another, making it suitable to independent component analysis and blind source separation.

- Maximization of Mutual Information between Output and Input (InfoMax): This is a supervised approach that seeks to convey the maximum information from the input through the mapping function to the output. It is similar to MaxEnt in applications.

- Maximization of Mutual Information between Output and Desired Signal (Information Filtering): This method is similar to InfoMax, but adapts the mapping function to make the output statistically similar to the any desired signal. This approach can be used in lieu of MSE for many supervised learning problems including function approximation, parameter estimation and pattern recognition Pri00].

- Maximization of Mutual Information between Output and Class Labels (ClassInfoFilt): This is a special case of information filtering described by Torkkola that seeks to group the outputs according to their class labels defined by the desired signal. This is most applicable to classification or class clustering to support classification [Tor00].

- Minimization of Error Entropy (MinErrEnt): This is a supervised approach that minimizes the entropy of the error between the output and a desired signal. This

approach is similar to information filtering and is applicable to most supervised learning problems [Erd00].

The principle drawback to using any of these information-theoretic criteria is the required computational demands of the implementation. Evaluation of these criteria requires the computation of the difference between two PDF's. The computational complexity of a straightforward implementation is proportional to $O(N^{M+2})$, where N is the number of samples in the dataset and M is the dimensionality of the random variable. By contrast, the computational complexity of the MSE is proportional to $O(N)$. As will be shown in the sections that follow, recent developments in ITL have resulted in a computationally tractable approach that reduces the complexity of the algorithm to be proportional to $O(N^2)$ [Pri00]. However, for large data sets, this still represents somewhat of a challenge.

Information Theoretic Learning Approach

Principe, Fisher, and Xu have shown that by using Renyi's quadratic entropy (RQE) in combination with Parzen's nonparametric PDF estimation, significant computational savings can be achieved when computing entropy-based optimization criterion. They realized similar computational savings for mutual information-based criterion by introducing two novel measures for PDF divergence based on the Cauchy-Schwartz inequality and Euclidean distance [Pri00].

The Parzen PDF estimator for a random variable, $\mathbf{Y} \in \mathcal{R}^M$, is a kernel-based method defined as,

$$\hat{f}_{\mathbf{Y}}(\mathbf{z},\mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N}\kappa(\mathbf{z}-y_i) \tag{3-10}$$

where N is the number of samples of the random variable, y_i are the observations of the

random variable, and κ is a kernel function. Although other choices exist, the basic ITL approach uses the symmetric Gaussian kernel described by (3-11).

$$\kappa(\mathbf{z}) = G(\mathbf{z}, \sigma^2\mathbf{I}) = \frac{1}{(2\pi\sigma^2)^{M/2}} e^{-\mathbf{z}^T\mathbf{z}/(2\sigma^2)} \tag{3-11}$$

Combining the kernel function from (3-11) with the Parzen estimator in (3-10) and substituting in (3-4) with $\alpha = 2$ results in the following estimator of Renyi's quadratic entropy for $\mathbf{Y}$ with observations $\{\mathbf{y}\}$,

$$\begin{aligned} H_{R2}(\mathbf{Y}) &= -\log V(\{\mathbf{y}\}) \\ V(\{\mathbf{y}\}) &= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} G(\mathbf{y}_i - \mathbf{y}_j, 2\sigma^2\mathbf{I}) \end{aligned} \tag{3-12}$$

where V is defined as the information potential (IP). Note that maximizing the entropy estimate, H_{R2}, in (3-12) is equivalent to minimizing the IP. For adaptation of the parameters in a mapping function, the sensitivities with respect to the weights are computed yielding,

where,

$$\begin{aligned} \frac{\partial}{\partial w} V(\mathbf{y}) &= \sum_{i=1}^{N}\left[\frac{\partial}{\partial \mathbf{y}_i} V(\mathbf{y})\right]\frac{\partial \mathbf{y}_i}{\partial w} = \sum_{i=1}^{N} \mathbf{F}_i^T \frac{\partial}{\partial w} g(w, \mathbf{x}_i) \\ \mathbf{F}_i &= \frac{\partial}{\partial \mathbf{y}_i} V(\mathbf{y}) = -\frac{1}{N^2\sigma^2}\sum_{j=1}^{N} G(\mathbf{y}_i - \mathbf{y}_j, 2\sigma^2\mathbf{I})(\mathbf{y}_i - \mathbf{y}_j), \end{aligned} \tag{3-13}$$

which lends itself to an adaptive training algorithm such as back-propagation.

Thus (3-12) shows that the combination of the Parzen PDF estimation and Renyi's quadratic entropy yields an entropy estimate that is based on the summation of interactions between each sample pair through the kernel function. As is evident, the computational complexity is proportional to $O(N^2)$. The major computational savings is due to the combination of the quadratic form of the PDF's in Renyi's quadratic entropy and the form of the Gaussian kernel selected for Parzen estimation.

For mutual information-based criterion, examination of (3-8) and (3-9) reveals that they are not quadratic in PDF's. This led Principe et al. to propose novel distance measures between two PDF's based on quadratic distance measurement. The first method, based on the Euclidean difference of vectors inequality, yields the following measure for quadratic mutual information (ED-QMI).

$$I_{ED}(\mathbf{X},\mathbf{Y}) = \iint p(x,y)^2\,dxdy + \iint p(x)^2\,p(y)^2\,dxdy - 2\iint p(x,y)p(x)p(y)dxdy \quad (3\text{-}14)$$

The second method, based on the Cauchy-Schwartz inequality, produces (3-15) as a measure for mutual information (CS-QMI).

$$I_{CS}(\mathbf{X},\mathbf{Y}) = \log\frac{\iint p(x,y)^2\,dxdy \iint p(x)^2\,p(y)^2\,dxdy}{\left(\iint p(x,y)p(x)p(y)dxdy\right)^2} \quad (3\text{-}15)$$

Both measures are nonnegative and reach zero when $p(x,y) = p(x)p(y)$ (i.e., when **X** and **Y** are independent) making their use appropriate for minimizing mutual information. For maximizing mutual information, Principe et al. show through examples that both measures can be used, however, ED-QMI is preferable because of the convex nature of the function.

By utilizing the criteria in (3-14) or (3-15) in combination with the Parzen PDF estimation from (3-10), the estimates described by (3-16) for the mutual information between two random variables, $\mathbf{Y}_1$ and $\mathbf{Y}_2$ with observations, $\mathbf{y} = [\mathbf{y}_{i1}, \mathbf{y}_{i2}]^T, i = 1,...N$, are obtained:

$$\begin{aligned} I_{CS}((\mathbf{Y}_1,\mathbf{Y}_2)\,|\,\mathbf{y}) &= \log\frac{V(\mathbf{y})V^1(\{\mathbf{y}_1\})V^2(\{\mathbf{y}_2\})}{V_{nc}(\mathbf{y})^2} \\ I_{ED}((\mathbf{Y}_1,\mathbf{Y}_2)\,|\,\mathbf{y}) &= V(\mathbf{y}) + V^1(\{\mathbf{y}_1\})V^2(\{\mathbf{y}_2\}) - 2V_{nc}(\mathbf{y}) \end{aligned} \quad (3\text{-}16)$$

where,

$$
\begin{aligned}
V(\mathbf{y}) &= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\prod_{l=1}^{2} G(\mathbf{y}_{il}-\mathbf{y}_{jl},2\sigma^2\mathbf{I}) \\
V^l(\mathbf{y}_j,\{\mathbf{y}_l\}) &= \frac{1}{N}\sum_{i=1}^{N} G(\mathbf{y}_{jl}-\mathbf{y}_{il},2\sigma^2\mathbf{I}), l=1,2 \\
V^l(\{\mathbf{y}_l\}) &= \frac{1}{N}\sum_{j=1}^{N} V^l(\mathbf{y}_j,\{\mathbf{y}_l\}), l=1,2 \\
V_{nc}(\mathbf{y}) &= \frac{1}{N}\sum_{j=1}^{N}\prod_{l=1}^{2} V^l(\mathbf{y}_j,\{\mathbf{y}_l\})
\end{aligned}
\tag{3-17}
$$

In (3-17), $V^l(\mathbf{y}_j,\{\mathbf{y}_l\})$ represents the partial marginal information potential for the sample $\mathbf{y}_j$ in its corresponding random variable (indexed by l). In (3-16) and (3-17) $V^l(\{\mathbf{y}_l\})$ is the l^{th} marginal information potential because it averages all the partial marginal information potentials for one index, l [Pri00]. The equations in (3-16) and (3-17) can support parameter adaptation by again taking the sensitivities with respect to the parameters.

Principe et al. have successfully applied this approach to obtain practical solutions to problems including independent component analysis, blind source separation, system identification, and function approximation. Despite these successes, several issues still hinder the widespread application of this approach. First, the information theoretic optimization criteria are highly dependent on the selection of the kernel size in the Parzen estimation of the PDF. Experimentation suggests that an adaptive kernel size, which assures sufficient interaction between the observations, produces favorable results. Second, although the computational complexity has been greatly reduced, even a complexity proportional to $O(N^2)$ becomes prohibitive for very large data sets.

Chapters 4, 5, and 6 address the reduction of computational complexity in using the ITL process for adaptive system training. These chapters focus on the information-

theoretic criteria based on RQE, CS-QMI and ED-QMI. These chapters address improving the computational efficiency in three ways:

1. Reduce the complexity of each criterion evaluation through algorithm optimization.
2. Reduce the number of training iterations required for convergence by applying advanced parameter search techniques.
3. Reduce the overall N^2 complexity by introducing a batch technique to partition the data.

CHAPTER 4
ALGORITHM OPTIMIZATION

Introduction

One of the principle drawbacks of using information-theoretic criteria for the training of learning systems is the extremely slow training due to the computational complexity of the criterion functions. In this chapter, the equations for entropy and mutual information estimation are examined from the perspective of computational efficiency. The optimization techniques of factorization, symmetry exploitation, and vectorization, are used to develop implementations of the ITL criteria with improved efficiency in terms of the number of floating-point operations (FLOPs) and computational time.

Baseline Entropy Algorithm

To establish a basis for comparison, a baseline algorithm implementation is used. This baseline algorithm results from a straightforward implementation of Renyi's quadratic entropy estimator, described in (4-1), without any attempts to simplify or optimize for efficiency. The following is obtained by combining (3-11) and (3-12),

$$H_{R2} = -\log\left\{\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{1}{\left(4\pi\sigma^2\right)^{M/2}}\exp\left(-\frac{\mathbf{d}_{ij}^T\mathbf{d}_{ij}}{4\sigma^2}\right)\right\} \tag{4-1}$$

where $\mathbf{d}_{ij} = (\mathbf{y}_i - \mathbf{y}_j)$. This straightforward, or brute force, implementation yields a computational complexity of roughly $N^2[4M+10]+NM+9$. Clearly, the dominant term is the factor of the quadratic term corresponding to N^2.

Using randomly generated data with various values for N and M, computational complexity metrics for the baseline implementation are tabulated in Tables 4-1, 4-2, and 4-3 under the columns labeled 'Baseline'. These metrics were collected using MATLAB version 5.3 on a personal computer with a 1Ghz Athlon processor. The baseline metrics provide a basis for comparison of various optimized algorithm implementations. It should be noted that for $M>1$, an additional MN^2 FLOPs is required because of MATLAB's specific implementation of the SUM command where the sum of K numbers requires K FLOPs versus the expected K-1. This issue, however, does not detract from the overall relevance of the discussion since it applies consistently to all compared cases where M is greater than one.

Entropy Algorithm Factorization

Factorization, or hoisting, is one of the most elementary techniques for algorithm optimization. The approach looks for repetitive operations that can be eliminated or reduced by pulling, or factoring, them out of major loops. Equation (4-1) is well suited for this type of optimization. Note that each evaluation of the kernel function contains the same divisor operand, $\left(4\pi\sigma^2\right)^{M/2}$, a six-FLOP operation assuming the σ^2 term has been previously computed and stored. Furthermore, the scaling factor, $4\sigma^2$, of the exponential is constant throughout the entire entropy computation and therefore need only be evaluated once. Since both of these terms contribute to the N^2 component of the complexity, their factorization can provide significant computational savings overall. These factorizations result in an entropy estimator described by (4-2).

$$H_{R2} = -\log\left\{\frac{1}{N^2\left(4\pi\sigma^2\right)^{M/2}}\sum_{i=1}^{N}\sum_{j=1}^{N}\exp\left(-\frac{\mathbf{d}_{ij}^T\mathbf{d}_{ij}}{C}\right)\right\}; C = 4\sigma^2 \qquad (4\text{-}2)$$

Another point to notice is that the division operation with the scaling factor, C, of the exponential is conducted a total N^2 times. Since the original data, $\mathbf{Y}$, is of dimension MxN, computations can be saved in the cases where $M<N$ by factoring out the scaling factor from the double summed exponential. This results in the following entropy estimator,

$$H_{R2} = -\log\left\{\frac{1}{N^2\left(4\pi\sigma^2\right)^{M/2}}\sum_{i=1}^{N}\sum_{j=1}^{N}\exp\left(-\mathbf{d}_{ij}^{*\,T}\mathbf{d}_{ij}^{*}\right)\right\} \tag{4-3}$$

where $\mathbf{d}_{ij}^{*} = (\mathbf{y}_i^{*} - \mathbf{y}_j^{*})$, and $\mathbf{Y}^{*}$ is computed by scaling the original input, $\mathbf{Y}$, by the square root of the scaling factor C, or 2σ. This factorization can provide additional computational savings particularly for large datasets where the number of samples, N, is much larger than the dimensionality, M.

Table 4-1. Algorithm optimization: Factorization.

Data Size		Complexity (FLOPs)			Computation Time (sec.)		
N	M	Baseline	Factorized	FRR	Baseline	Factorized	TRR
50	1	35059	15115	2.3	0.13	0.07	1.8
50	2	50109	30215	1.7	0.17	0.09	1.8
50	3	62659	42815	1.5	0.18	0.09	1.9
100	1	140109	60215	2.3	0.51	0.28	1.8
100	2	200209	120415	1.7	0.66	0.37	1.8
100	3	250309	170615	1.5	0.68	0.37	1.8
200	1	560209	240415	2.3	2.06	1.13	1.8
200	2	800409	480815	1.7	2.61	1.50	1.7
200	3	1000609	681215	1.5	2.72	1.48	1.8
500	1	3500509	1501015	2.3	12.80	7.12	1.8
500	2	5001009	3002015	1.7	16.42	9.30	1.8
500	3	6251509	4253015	1.5	17.03	9.32	1.8
1000	1	14001009	6002015	2.3	51.19	28.45	1.8
1000	2	20002009	12004015	1.7	65.91	37.13	1.8
1000	3	25003009	17006015	1.5	68.27	37.46	1.8
2000	1	56002009	24004015	2.3	208.01	112.93	1.8
2000	2	80004009	48008015	1.7	265.73	147.91	1.8
2000	3	100006009	68012015	1.5	271.06	149.62	1.8

Table 4-1 compares the results of this implementation with the baseline approach. The table shows the average computation times and number of FLOPs required to complete one evaluation of the entropy and the associated information forces for both implementations. It also shows the FLOP-count-reduction-ratio (FRR) and the computational-time-reduction-ratio (TRR) for the factorized implementation versus the baseline. The resulting computational complexity of the factorized implementation is roughly $N^2[4M+2]+N[M+1]+15$, thus reducing original complexity by roughly $8N^2$ operations. For M=1 this is more than a 50 percent reduction in the number of FLOPs.

Exploitation of Symmetry

Another method for improving the computational efficiency of an algorithm is to look for symmetries or simplifications in the algorithm structure to eliminate redundant or similar calculations. Again, the ITL entropy criterion lends itself to this type of optimization. By noting the fact that $V_{ij} = -V_{ji}$, roughly half of the kernel function evaluations can be omitted. In addition, when the kernel function is factorized as above, its evaluation in the cases where i=j results in $V_{ii} = 1$ and $F_{ii} = 0$. Therefore, the computation of the diagonal terms can also be greatly simplified. Equation (4-4) describes the symmetric implementation with factorization.

$$H_{R2} = -\log\left\{\frac{2}{N^2\left(4\pi\sigma^2\right)^{M/2}}\left(\frac{N}{2}+\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\exp\left(-\mathbf{d}_{ij}^{*T}\mathbf{d}_{ij}^{*}\right)\right)\right\} \tag{4-4}$$

This symmetric, factorized implementation results in a computational complexity of roughly $N^2[2.5M+1]-N[1.5M-1]+21$. Table 4-2 shows the computational reductions achieved by this method for various values of N and M. Note that the

exploitation of symmetry yields even greater reductions in both the complexity and computational time metrics than with factorization alone.

Table 4-2. Algorithm optimization: Symmetry with factorization.

Data Size		Complexity (FLOPs)			Computation Time (sec.)		
N	*M*	Baseline	Symmetric	FRR	Baseline	Symmetric	TRR
50	1	35059	8746	4.0	0.13	0.05	2.8
50	2	50109	17421	2.9	0.17	0.06	2.9
50	3	62659	24871	2.5	0.18	0.06	3.1
100	1	140109	34971	4.0	0.51	0.18	2.9
100	2	200209	69821	2.9	0.66	0.23	2.9
100	3	250309	99721	2.5	0.68	0.23	2.9
200	1	560209	139921	4.0	2.06	0.70	2.9
200	2	800409	279621	2.9	2.61	0.92	2.8
200	3	1000609	399421	2.5	2.72	0.92	2.9
500	1	3500509	874771	4.0	12.80	4.43	2.9
500	2	5001009	1749021	2.9	16.42	5.77	2.8
500	3	6251509	2498521	2.5	17.03	5.79	2.9
1000	1	14001009	3499521	4.0	51.19	17.74	2.9
1000	2	20002009	6998021	2.9	65.91	22.96	2.9
1000	3	25003009	9997021	2.5	68.27	23.35	2.9
2000	1	56002009	13999021	4.0	208.01	71.07	2.9
2000	2	80004009	27996021	2.9	265.73	92.00	2.9
2000	3	100006009	39994021	2.5	271.06	92.66	2.9

Entropy Vectorization

The methods of factorization and symmetry exploitation achieve improvements in computational efficiency by reducing the overall number of operations required to compute a given metric. Vectorization, on the other hand, improves efficiency by taking advantage of modern software and processor design features. Most processors today are optimized to perform single-instruction multiple-data (SIMD) and vector operations very efficiently. Many software implementation environments and programming languages are optimized to make use of these processor characteristics. These features can be exploited to greatly reduce computation times by developing algorithm implementations that operate on vector and matrix operands. Once again, the ITL criteria are well suited

for optimization in this manner. The second indexed summation in (4-4) can be replaced using standard vectorization techniques to remove the need for the *j*-indexing. By describing the basic operand for this summation as the vector in (4-5), appropriate vectorized exponential and summation functions (such as those resident in MATLAB) can be applied to speed up computational time.

$$\mathbf{D}_i^* = [\mathbf{d}_{i,i+1}^{*\,T}\mathbf{d}_{i,i+1}^* \ldots \mathbf{d}_{i,N}^{*\,T}\mathbf{d}_{i,N}^*] \tag{4-5}$$

Table 4-3. Algorithm optimization: Vectorized with symmetry and factorization.

Data Size		Complexity (FLOPs)			Computation Time (sec.)		
N	*M*	Baseline	Vector	FRR	Baseline	Vector	TRR
50	1	35059	8838	4.0	0.13	0.007	19.1
50	2	50109	17561	2.9	0.17	0.008	21.6
50	3	62659	25059	2.5	0.18	0.008	22.3
100	1	140109	35163	4.0	0.51	0.016	32.3
100	2	200209	70111	2.9	0.66	0.018	37.7
100	3	250309	100109	2.5	0.68	0.019	36.2
200	1	560209	140313	4.0	2.06	0.041	50.7
200	2	800409	280211	2.9	2.61	0.049	52.8
200	3	1000609	400209	2.5	2.72	0.070	38.7
500	1	3500509	875763	4.0	12.80	0.195	65.6
500	2	5001009	1750511	2.9	16.42	0.272	60.4
500	3	6251509	2500509	2.5	17.03	0.366	46.6
1000	1	14001009	3501513	4.0	51.19	0.830	61.7
1000	2	20002009	7001011	2.9	65.91	1.175	56.1
1000	3	25003009	10001009	2.5	68.27	1.511	45.2
2000	1	56002009	14003013	4.0	208.01	3.658	56.9
2000	2	80004009	28002011	2.9	265.73	5.022	52.9
2000	3	100006009	40002009	2.5	271.06	6.272	43.2

The computational complexity of the vectorized implementation is approximately $N^2[2.5M+1] - N[0.5M-2] - 2M + 15$. This complexity measure is roughly the same as in the symmetric-factorized case. However, the simple replacement of the indexed exponentials and summations with their vector equivalents yields drastic improvements in computation times as shown in Table 4-3. Note that for large sample sizes, the computational time has been reduced by factors of 60 or more in some cases. Figure 4-1

shows graphically the computational efficiencies gained via optimization for the entropy estimator in the case of M=2.

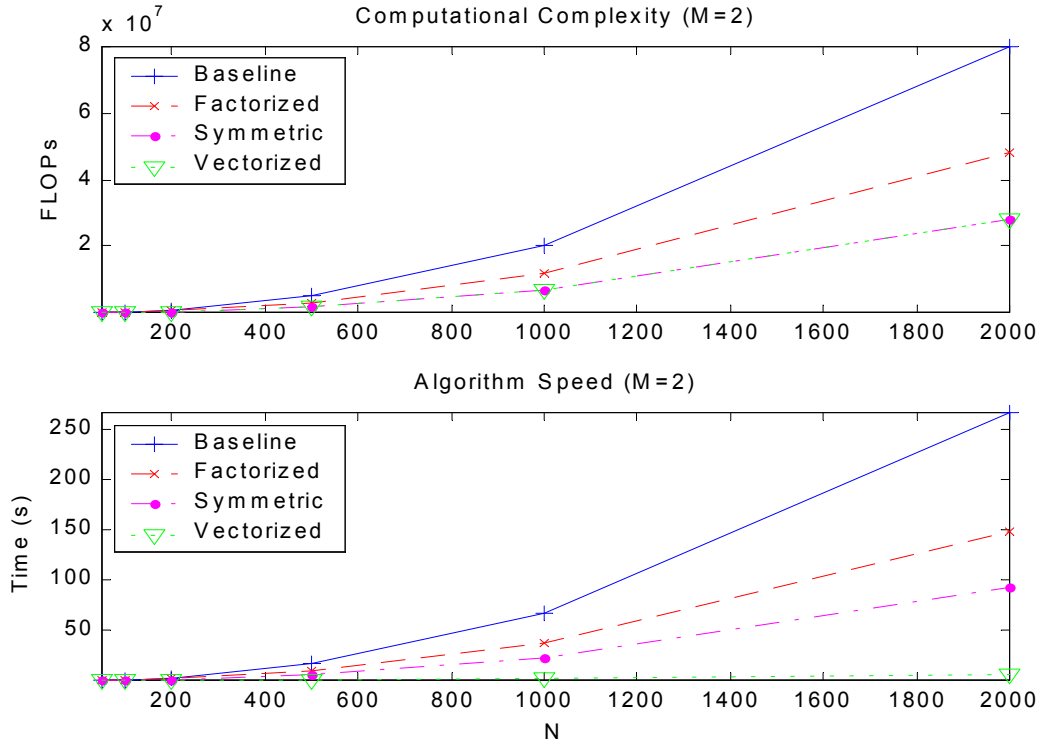

Figure 4-1. Entropy estimator optimization results.

Results for the QMI Criterion

The same techniques for algorithm optimization were applied to (3-16) and (3-17) for ED-QMI and CS-QMI, respectively. Table 4-4 compares the resulting computational metrics for a baseline ED-QMI implementation with one that has been optimized using factorization, symmetry, and vectorization. In these cases, the QMI is evaluated between an M-dimensional, N-sample data set and a one-dimensional, N-sample data set. The results obtained for CS-QMI are nearly identical as those for ED-QMI and are not shown for brevity. Equation (4-6) summarizes a factorized and symmetric version of (3-17), where $\mathbf{d}_{ijl}^{*} = (\mathbf{y}_{il}^{*} - \mathbf{y}_{jl}^{*})$. The operands in this representation were also vectorized as in the entropy case to yield significant savings in computation time.

$$V(\mathbf{y}) = \frac{2}{N^2\left(4\pi\sigma^2\right)^{M/2}}\left(\frac{N}{2} + \sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\prod_{l=1}^{2}\exp\left(-\mathbf{d}_{ijl}^{*T}\mathbf{d}_{ijl}^{*}\right)\right)$$
$$V^{l}(\{\mathbf{y}_l\}) = \frac{2}{N^2\left(4\pi\sigma^2\right)^{M_l/2}}\left(\frac{N}{2} + \sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\exp\left(-\mathbf{d}_{ijl}^{*T}\mathbf{d}_{ijl}^{*}\right)\right) \quad (4\text{-}6)$$

Table 4-4. Algorithm optimization: ED-QMI baseline and optimized results.

Data Size		Complexity (FLOPs)			Computation Time (sec.)		
N	*M*	Baseline	Vector	FRR	Baseline	Vector	TRR
50	1	88071	34345	2.6	0.48	0.028	17.3
50	2	136179	56125	2.4	0.59	0.038	15.6
50	3	184338	77956	2.4	0.71	0.039	18.3
100	1	351121	136170	2.6	1.94	0.054	35.9
100	2	542329	222225	2.4	2.49	0.076	32.8
100	3	733638	308381	2.4	2.84	0.100	28.4
200	1	1402221	542320	2.6	7.63	0.170	44.9
200	2	2164629	884425	2.4	9.94	0.220	45.2
200	3	2927238	1226731	2.4	11.37	0.330	34.5
500	1	8755521	3380770	2.6	48.28	0.830	58.2
500	2	13511529	5511025	2.5	60.26	1.420	42.4
500	3	18268038	7641781	2.4	71.51	2.030	35.2
1000	1	35011021	13511520	2.6	191.20	4.120	46.4
1000	2	54023029	22022025	2.5	241.73	5.980	40.4
1000	3	73036038	30533531	2.4	289.07	8.020	36.0
2000	1	140022021	54023020	2.6	772.26	15.870	48.7
2000	2	216046029	88044025	2.5	966.42	24.060	40.2
2000	3	292072038	122067031	2.4	1153.88	33.940	34.0

Conclusions

Straightforward implementations of ITL criteria have produced computationally expensive and time consuming results. By applying some basic principles of algorithm optimization, significant savings in both measures have been demonstrated. The combination of factorization, symmetry exploitation, and vectorization has reduced the algorithm complexity in terms of number of operations by as much as 75%. Similarly, the required computation time has been reduced by as much as 98%. A case in point is the 2000-sample quadratic entropy evaluation that originally took over four minutes has now been reduced to roughly five seconds. Although the complexity of the algorithm

remains $O(N^2)$, the multiple of the quadratic term has been greatly reduced, thereby increasing the practical utility of ITL methods.

CHAPTER 5
ADVANCED PARAMETER SEARCH TECHNIQUES

Introduction

Parameter optimization for learning systems generally requires the use of iterative techniques since analytical solutions exist only for special cases of both topology and criteria [Bis95]. These iterative techniques, which search the parameter space of the mapping function in an attempt to achieve a desired or optimum performance level, often require many epochs in order to converge to an acceptable level of performance. When these techniques are applied using information-based criteria to determine system performance, a large number of iterations can quickly render their usage to be computationally prohibitive. This dilemma motivates the investigation of advanced parameter search algorithms for ITL in order to arrive at a desired performance level more efficiently. This chapter addresses the adaptation and application of some familiar advanced parameter search algorithms to the training of systems using ITL criteria. Several examples are used to show the benefits of applying advanced training techniques to ITL problems.

Baseline Technique: Gradient Descent (GD)

The most prevalent and basic parameter search technique for learning system adaptation is the gradient descent algorithm. The GD algorithm essentially follows the slope, or gradient, of the performance surface towards a minimum. Although simple and effective, GD is often slow to converge to a desired performance level and frequently requires hundreds and even thousands of iterations. Despite and perhaps because of its

simplicity, the training of ITL systems has been done almost exclusively with GD learning. GD uses the following update rule,

$$\Delta w = -\eta \nabla P \tag{5-1}$$

where Δw represents the change to the learning system parameter vector, or weights, ∇P represents the gradient of the performance surface, and η is the step size or learning rate. The convergence of the GD algorithm depends strongly on the appropriate selection of the learning rate. Too large a learning rate can cause oscillation or divergence, while too small a learning rate leads to slow adaptation and the need for many epochs. The GD method will serve as the baseline for comparison of other more advanced parameter search techniques.

Advanced Search Techniques

The general problem of parameter optimization has been the focus of much research and has motivated the development of many advanced techniques for parameter search [Bis95, Dem00, Hag96]. In the paragraphs below, several of these advanced search methods are described in anticipation of their application to training of ITL systems.

Improved GD Algorithms

Many improvements have been developed to increase the efficiency of the basic GD approach. Two significant improvements have been the incorporation of an adaptive learning rate and the addition of a momentum term to the basic GD algorithm. The adaptive learning rate gradient descent (GDA) algorithm monitors the current performance of the learning system and adapts the learning rate as follows: if performance improves, learning rate is increased slightly; if performance worsens, the

learning rate is decreased moderately and the latest update to the system parameters is discarded. This allows the learning rate to adjust based on the local complexity of the performance surface. The learning rate adaptation can be summarized by (5-2).

$$\eta_{k+1} = \begin{cases} a\eta_k & if \quad P_k < P_{k-1} \quad where \quad a > 1 \\ b\eta_k & if \quad P_k > P_{k-1} \quad where \quad b < 1 \end{cases} \tag{5-2}$$

The GDA algorithm thus reduces the training performance dependency on the sometimes ad hoc selection of the learning rate.

The addition of a momentum term to GD serves as a low pass filter for parameter adaptation. This allows parameter adaptation to avoid some levels of noise and shallow local minima in the performance surface. Adaptation is taken in the direction of a weighted sum of the current GD direction and the last step taken. The following summarizes the GD with momentum (GDM) update rule,

$$\Delta w_k = (1-\alpha)(-\eta \nabla P) + \alpha \Delta w_{k-1} \tag{5-3}$$

where α is the momentum constant. Clearly, the momentum constant determines how much credence is given to the historical trend versus the current estimate. Another useful method that combines GD with adaptive learning rate and momentum (GDX) will be studied as well. Note that both GDA and GDX assume a static performance surface that allows epoch-to-epoch comparison of performance in order to determine the learning rate adaptation.

Resilient Backpropagation (RP)

Resilient backpropagation was introduced to help eliminate some of the problems encountered when training neural networks containing saturating nonlinearities using GD methods [Dem00, Rie93]. RP uses the sign of the gradient, rather than the actual gradient

to determine the direction of the weight change. This helps to improve the slow learning caused by the near-zero gradients in the saturated regions of the nonlinear processing elements. The magnitudes of the weight changes are adapted according to the direction of the most recent epochs as follows: if a weight change is in the same direction for the last two epochs, the magnitude of the change is increased; if the update direction is different, the magnitude of change is decreased [Rie94]. The update rule for RP is given as,

$$\Delta w_k^{ij} = -sign(\nabla P_k^{ij})\delta_k^{ij} \tag{5-4}$$

where the magnitude, δ_k^{ij}, is adapted as,

$$\delta_k^{ij} = \begin{cases} a\delta_{k-1}^{ij} & if \quad \nabla P_k^{ij} * \nabla P_{k-1}^{ij} > 0 \quad where \quad a > 1 \\ b\delta_{k-1}^{ij} & if \quad \nabla P_k^{ij} * \nabla P_{k-1}^{ij} < 0 \quad where \quad b < 1 \end{cases} \tag{5-5}$$

RP has proven to be a robust algorithm since its adaptation is governed more by the ongoing adaptive behavior of the weights than the shape of the performance surface.

Conjugate Gradient Algorithms

Conjugate gradient algorithms were developed in order to select a more efficient search direction than the standard GD approach. These algorithms begin with a steepest descent step and then choose conjugate directions for adaptation rather than steepest descent. For quadratic performance surfaces, conjugate directions form a complete basis in the parameter space and generally provide much faster convergence than GD directions [Bis95]. Most conjugate gradient methods determine the change magnitude using a line search technique. The basic update for conjugate gradient methods is,

$$\Delta w_k = \alpha_k \mathbf{d}_k \tag{5-6}$$

$$\mathbf{d}_k = -\nabla P_k + \beta_k \mathbf{d}_{k-1} \tag{5-7}$$

where α_k is determined using a line search, $\mathbf{d}_k$ represents the conjugate search direction, and β_k determines the method in which the next conjugate direction is chosen. The Fletcher-Reeves (CGF) method chooses the direction with,

$$\beta_k = \frac{\nabla P_k^T \nabla P_k}{\nabla P_{k-1}^T \nabla P_{k-1}} \tag{5-8}$$

while the Polak-Ribiere (CGP) updates the direction with,

$$\beta_k = \frac{(\nabla P_{k-1} - \nabla P_{k-2})^T \nabla P_k}{\nabla P_{k-1}^T \nabla P_{k-1}} \tag{5-9}$$

For all conjugate gradient algorithms, the search direction is periodically reset to the negative of the gradient. Typically, this is done when the number of epochs equals the number of parameters to be optimized. The Powell-Beale (CGB) uses a reset condition based on a measure of orthogonality [Pow77]. The CGB method also computes the search direction from a linear combination of the negative gradient, the previous search direction, and the last search direction before the previous reset [Dem00].

The Scaled Conjugate Gradient (SCG) method was developed to eliminate the computationally expensive line search from the conjugate gradient approach [Dem00, Mol93]. The method takes advantage of computational savings when the product of the Hessian and a vector are computed. However, the SCG is a model-trust-region method and requires the addition of a scaling coefficient, λ, to govern the trust region. The basic update takes the form of (5-6) and (5-7) with,

$$\alpha_k = -\frac{\mathbf{d}_k^T \nabla P_k}{\mathbf{d}_k^T \mathbf{H}_k \mathbf{d}_k + \lambda_k \|\mathbf{d}_k\|^2} \tag{5-10}$$

$$\beta_k = \frac{\nabla P_k^T \nabla P_k - \nabla P_k^T \nabla P_{k-1}}{-\mathbf{d}_{k-1}^T \nabla P_k} \tag{5-11}$$

where λ is adjusted to ensure the validity of the model. All the conjugate gradient methods assume a static performance surface to properly execute the line search or to determine whether a performance improvement can be realized in the case of SCG.

Newton's Method

Newton's method makes use of the second derivative information, via the Hessian matrix, to arrive at the performance minimum in fewer steps. The basic update for Newton's method is given by,

$$\Delta w = -\mathbf{H}^{-1}\nabla P \tag{5-12}$$

However, since the direct computation of the Hessian is computationally expensive, quasi-Newton methods have been developed that iteratively estimate the inverse of the Hessian at each epoch. One of the most successful approaches is the Broyden, Fletcher, Goldfarb, and Shanno (BFGS) quasi-Newton algorithm [Dem00]. Quasi-Newton approaches determine the change direction using the inverse Hessian estimate and the change magnitude using a line search as follows,

$$\Delta w = -\alpha\hat{\mathbf{H}}^{-1}\nabla P \tag{5-13}$$

where α is determined using a line search. The BFGS method requires storage of the inverse Hessian approximation, which may consume prohibitive amounts of memory for large networks.

The One-Step Secant (OSS) algorithm is a simplified quasi-Newton approach that was developed to avoid the storage requirements for the Hessian estimate [Bat92, Dem00]. The method assumes that for each epoch the previous Hessian estimate was the identity matrix and uses the same update rule as (5-13). Each of these methods assumes a static performance surface for proper execution of their line searches.

Levenberg-Marquardt (LM) Algorithm

The Levenberg-Marquardt algorithm uses a model-trust-region technique specifically designed to minimize a sum of squares error (SSE) function. The model assumes a locally linear network, which produces a parabolic error surface [Bis95, Dem00]. Based on these assumptions, the algorithm computes estimates of the Hessian matrix and gradient as,

$$\mathbf{H} = \mathbf{J}^T\mathbf{J} \tag{5-14}$$

$$\nabla P = \mathbf{J}^T\mathbf{e} \tag{5-15}$$

where **J** is the Jacobian matrix and **e** is the vector of network errors computed as the difference between the desired and current outputs. The method adds a variable diagonal element, μ, to the Hessian approximation to compensate when the model-trust assumptions are not valid. The value of μ is decreased with each successful step and increased with each unsuccessful step. As μ approaches zero, the weight adaptation direction becomes that of Newton's method. For large μ, the direction becomes parallel to steepest descent. The LM update rule is summarized by,

$$\Delta w = -\left[\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}\right]^{-1}\mathbf{J}^T\mathbf{e} \tag{5-16}$$

The LM algorithm assumes a static performance surface for divergence measures and its efficiency is highly dependent upon the assumptions regarding the error criterion.

Performance Divergence Measures

Some of the algorithms above use a performance divergence measure to decide whether the new weight update should be kept or whether it should be discarded while readjusting the parameters of the search routine. The simplest test compares the current performance value to the previous performance value and is summarized by,

$$P_{k+1} > P_k \tag{5-17}$$

If the test in (5-17) is false, it is considered positive for learning and true is considered divergent. However, to increase robustness for noise and shallow local minima, a small amount of performance degradation is often allowed. A common implementation of this measure is the specification of a small maximum change percentage in the performance. For MSE and SSE criterion, this is typically implemented by the evaluation of (5-18) for the case of a five percent degradation tolerance.

$$\frac{P_{k+1}}{P_k} > 1.05 \tag{5-18}$$

If the test in (5-18) evaluates to true, the performance is considered to be diverging beyond the tolerable limit, the current update is discarded, and the search parameters are adjusted to stabilize learning accordingly. If it evaluates false, adaptation is continued using either the current search parameters or parameters adjusted to speed up learning.

Line Search Routines

Several of the algorithms use line search routines to determine the adaptation step size in a particular direction for each epoch. A line search routine essentially looks for the minimum performance value along a line in a given direction in parameter space. Many line search routines exist including bisection, quadratic interpolation, cubic interpolation, golden section search and backtracking search. In addition, several hybrid line search routines are popular including Brent's search, hybrid bisection-cubic search, and Charalambous' search. Regardless of the search method, each assumes a static performance surface as performance values are evaluated along the search line and used to determine the next estimate towards the minimum.

Difficulties and Remedies for ITL

In general, straightforward application of most of the advanced parameter search techniques described above to ITL systems will not yield acceptable results. Essentially, the difficulties are due to assumptions in these algorithms that do not necessarily hold true for ITL systems in general. The paragraphs below examine some of the characteristics of ITL systems that necessitate the careful tailoring of most of these routines to accommodate ITL criteria. In addition, proposed remedies are suggested for each case. This section concludes with a summary of how the various advanced training algorithms have been adapted for use with ITL systems. The Appendix describes the tailoring process for one of the simplest (GD) and one of the most complex (LMLS) training algorithms in more detail.

Adaptive Kernel Size

A key component of PDF estimation is the appropriate selection of the kernel size. For ITL adaptation, the kernel size is also the impetus for learning since adaptation is caused by interactions between samples, which are directly governed by their spacing and the kernel size. Research has demonstrated that allowing the kernel size to adapt during training can lead to improved results [Erd02]. However, this characteristic of ITL systems can present problems for many of the advanced parameter search techniques. The cause of the problems is that the kernel size not only governs the level of interaction between samples, but also and consequently, controls the shape of the performance surface.

Figure 5-1 compares the performance surface for several values of kernel size. The figure shows the ED-QMI performance surface for three different values of kernel size (σ^2=0.001, 0.01, 0.1) for Example 3 described below. The figures show that

although the surfaces generally have similar shapes and similar location of extrema within the weight space, there are two significant differences. First, as intuition suggests, the relative smoothness of the performance surface increases with larger kernel size. Notice the shallow local minima for in the case when σ^2=0.001. Secondly, the value of the performance function varies with the kernel size as shown by the different scales for the performance surface plots on the right.

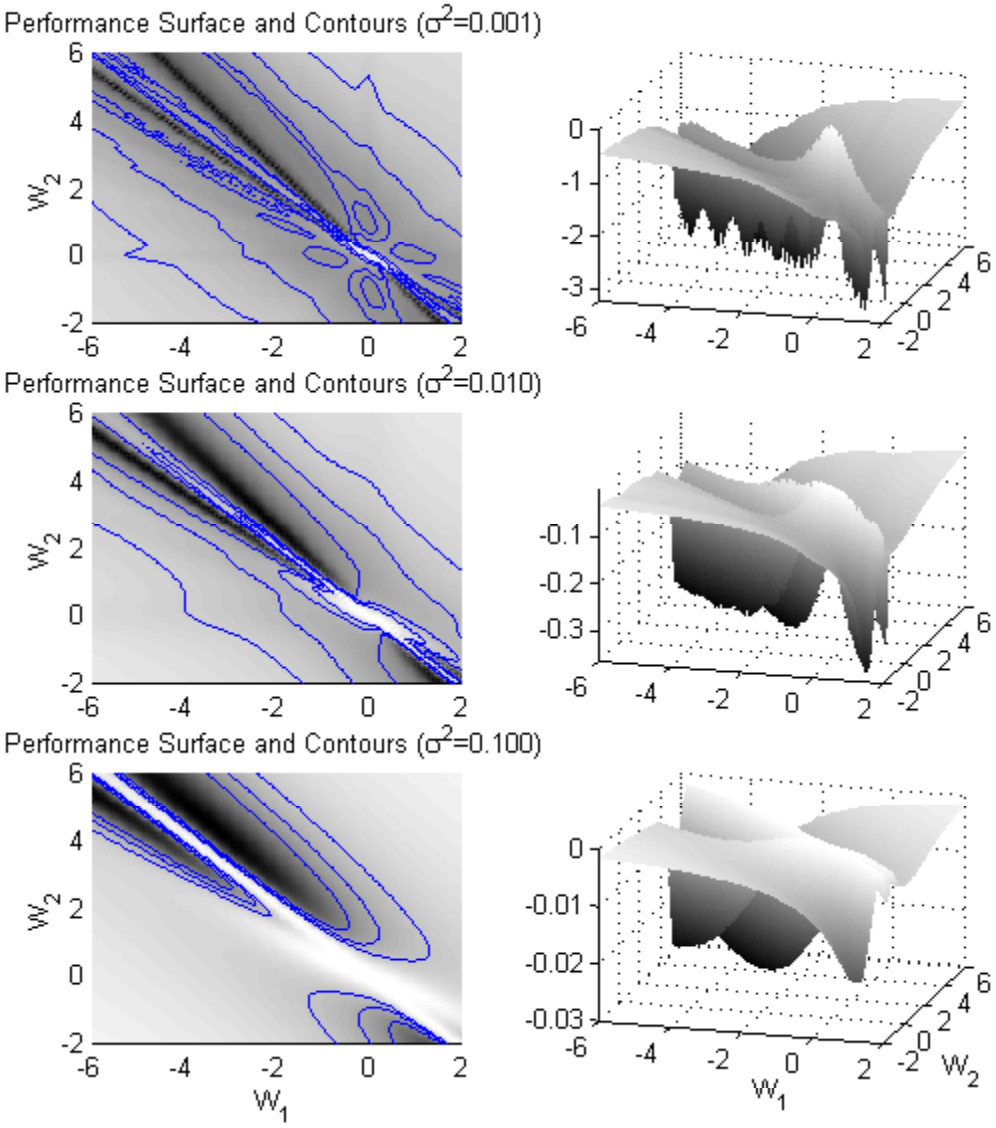

Figure 5-1. Performance surface variation with kernel size.

These differences need to be carefully considered when training ITL systems and especially when comparing the training performance of these systems. The impact of a dynamic performance surface on a search algorithm that assumes a static surface can be disastrous, quickly leading to divergence and instability. Some of the key methods that are impacted include performance divergence measures, line searches, and conjugate direction selection.

In order to accommodate a dynamic performance surface while taking advantage of the improved efficiency of advanced training techniques, some limitations are imposed on the adaptation of the kernel size. The kernel size is frozen at its current value during the execution of any sequence of search algorithm steps that require a performance comparison or a line search execution. Once these steps have been complete to an end and the resulting set of weights is determined, kernel size adaptation is allowed to take place. For many algorithms, this requires re-evaluation of the performance so that it can be used in the next epoch as the basis for comparison. Although this results in an increased computational burden for each epoch in a given algorithm, the advantage of adapting the kernel size based on the new distribution of samples is realized.

Positive and Negative Performance Values

Use of the MSE and SSE criteria produce a nonnegative performance surface where a zero value corresponds to zero error, or a perfect fit. The nature of Renyi's quadratic entropy estimator from (3-12) allows the entropy to take on any real value. This characteristic can be troublesome to some of the performance divergence tracking measures described above. Equation (5-18) no longer works if both performance values are not positive. A simple solution is to adjust (5-18) to accommodate negative numbers as described by (5-19).

$$\frac{P_{k+1} - P_k}{|P_k|} + 1 > 1.05 \tag{5-19}$$

While (5-19) works in practice, it is not without as robust as might be expected. Notice that this solution, like (5-18), allows a smaller absolute change as the performance approaches zero, but in this case, from both the positive and negative sides. This is shown in Figure 5-2. While this type behavior is desirable for SSE and MSE criterion since zero is the ideal terminal performance, the zero value has no such significance when evaluating the entropy criterion. This behavior likely would render searches to be less robust to noise as the performance was crossing this boundary.

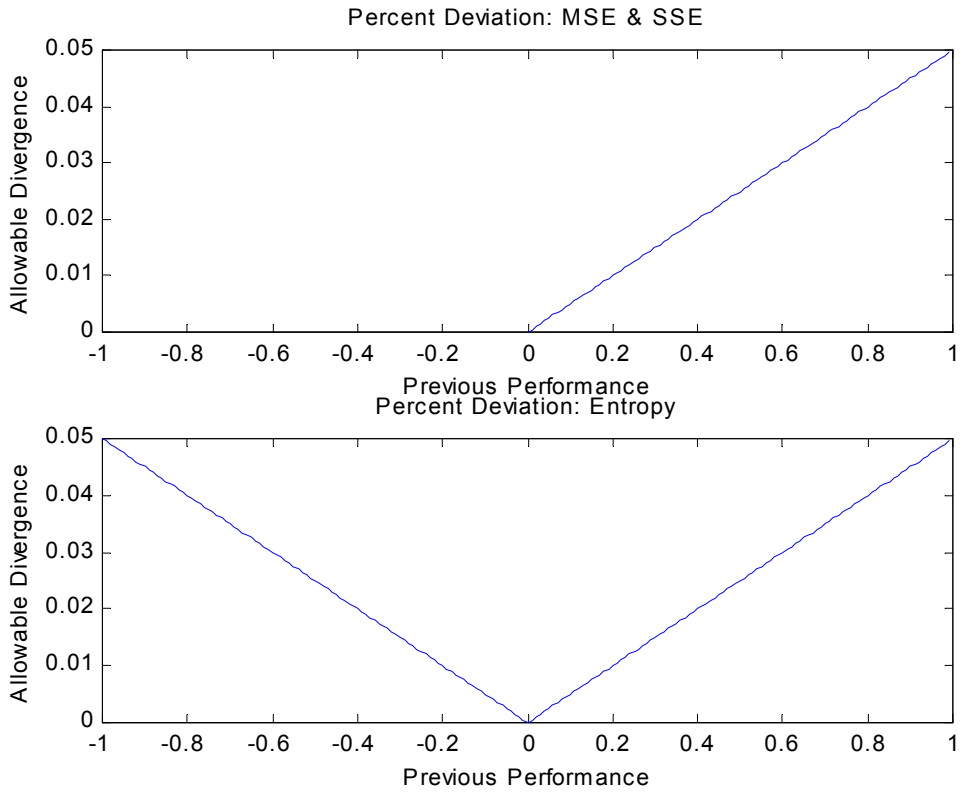

Figure 5-2. Performance divergence measure.

This shortcoming motivates the search for a performance divergence metric that is tuned to the specific characteristics of entropy estimation. To arrive at such a metric, first

note that for a given set of entropy estimation parameters (N, M, σ), where N is the number of samples, M is the dimensionality of the sample space, and σ is the kernel size for the PDF estimation, the maximum and minimum values of entropy are readily obtained using,

$$H_{\min} = \frac{M}{2}\log 4\pi + \frac{M}{2}\log\sigma^2 \tag{5-20}$$

$$H_{\max} = \frac{M}{2}\log 4\pi + \frac{M}{2}\log\sigma^2 + \log N \tag{5-21}$$

This suggests that at any point during training, the current entropy estimate can be compared relative to the maximum range of entropy using the metric in (5-22).

$$H_{relative} = \frac{H_{current} - H_{\min}}{H_{\max} - H_{\min}} \tag{5-22}$$

Equation (5-22) is defined as the relative entropy and has a range of [0,1]. The expressions in (5-20), (5-21), and (5-22) are discussed in further detail in Chapter 7. At this point, it is sufficient to note that the relative entropy provides a basis for a performance divergence metric that is indeed tuned to the characteristics of our entropy estimator. Consider the test described by (5-23).

$$H_{relative(k)} - H_{relative(k+1)} > 0.05 \tag{5-23}$$

Equation (5-23) provides the desired metric for the MaxEnt criterion.

Relative Error versus Absolute Squared-Error

While the MSE and SSE criteria provide an absolute error measure, ITL criteria provide only a relative measure of error. The LM algorithm, designed specifically for the SSE criterion, uses the assumption of a quadratic performance surface along with the absolute SSE to determine both the direction and magnitude of weight change. This

presents two problems. First, the computation of the SSE criterion is typically built into LM algorithm implementations and requires a set of desired target outputs to compute the absolute SSE. For ITL training, a desired target set is not always available and even if one is available, it need not have the same dimensionality as the output space. This difficulty can be overcome relatively easily by substituting the ITL information forces from (3-13) for the error term, **e**, in the gradient computation of the LM algorithm shown in (5-15).

Although this approach now provides the correct search direction according to the ITL criterion, it introduces a second problem. LM uses the absolute error assumption to determine the magnitude of the step size; however, in ITL systems the information forces only provide a relative measure of the error for each output sample. This mismatch in assumptions can lead to improper step sizes during LM parameter adaptation as shown in Figure 5-3. The figure shows the contours of an ITL performance surface for a simple two-parameter system identification problem of an infinite impulse response (IIR) filter with N=40 samples. In addition, the weight tracks for three implementations of LM-ITL algorithms are also shown. The first, labeled 'LM', is the weight track for the implementation describe above. Although the algorithm seems to be pointing in the appropriate direction at each step, many epochs are required for convergence compared to typical LM behavior because of an inappropriately small step size. To help alleviate this problem, two methods are proposed.

The first method, shown in the figure as 'LMSC', simply scales the step size by scaling the ITL information forces in proportion to the number of samples, N, in the ITL problem. This approach assumes that the ITL information forces provide an average

error that needs to be scaled in order to yield a larger magnitude for larger numbers of samples. An ad hoc value for the scale factor of $N/9$ was determined to be fairly robust empirically. As is shown in the figure, this approach takes larger steps than LM and arrives at the minimum in fewer epochs. The second approach combines LM with a line search algorithm (LMLS) to determine the step magnitude. Although the line search adds computational complexity per epoch, convergence happens in very few steps. The update rule for this method is given by,

$$\Delta w = -\alpha\left[\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}\right]^{-1}\mathbf{J}^T\mathbf{e} \tag{5-24}$$

where α is determined using a line search.

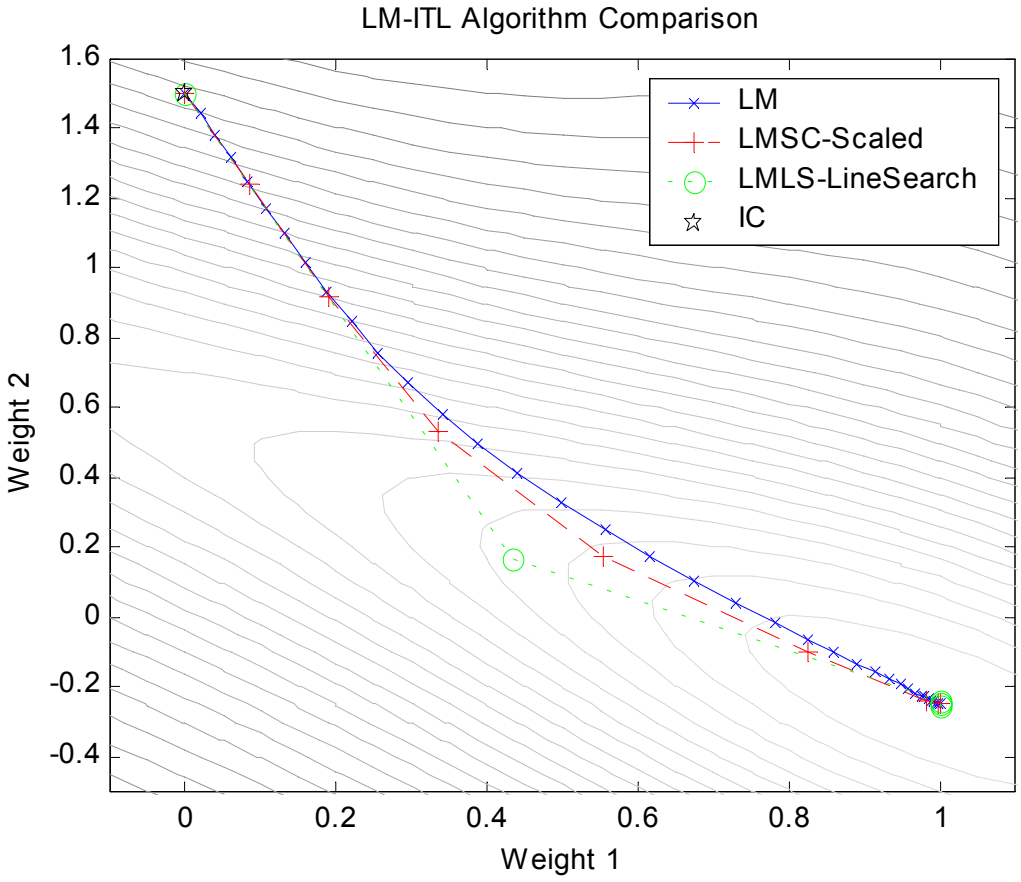

Figure 5-3. Levenberg-Marquardt ITL algorithms: Weight tracks.

The computational complexity of these three methods is compared in Figure 5-4. Notice how both of the proposed methods converge more efficiently than the basic LM method. This improved efficiency becomes more pronounced for larger values of *N*.

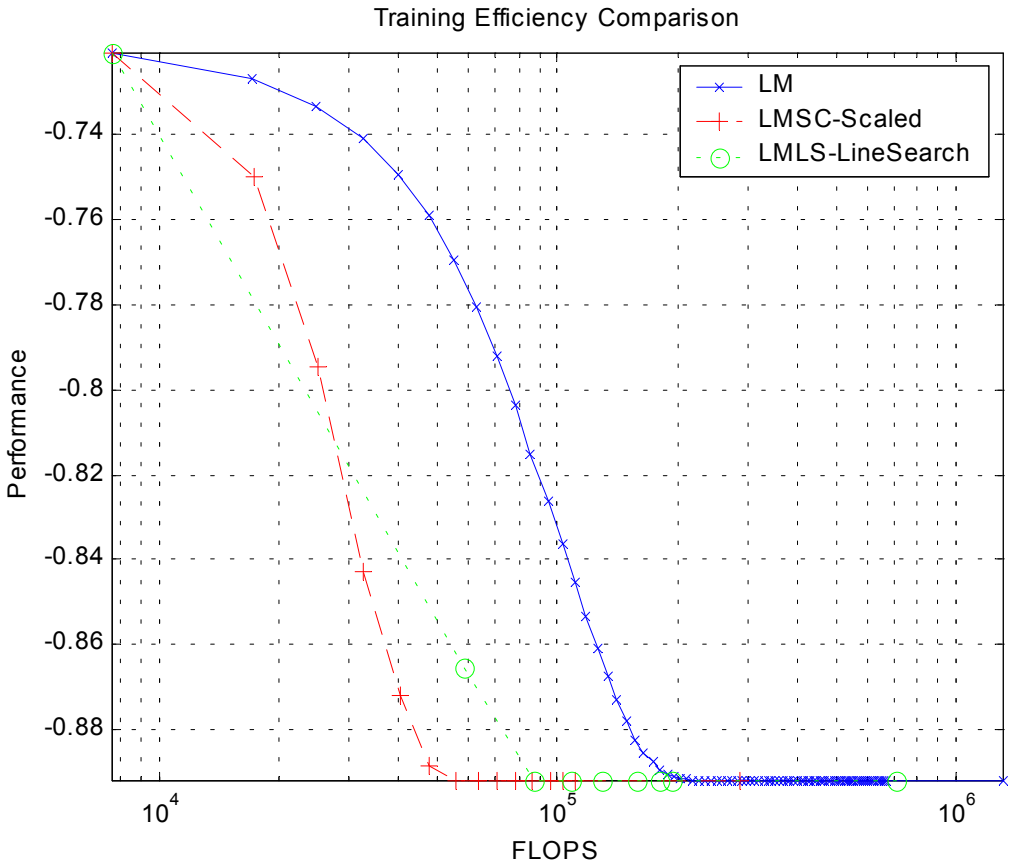

Figure 5-4. Levenberg-Marquardt ITL algorithms: Training efficiency.

Summary of Advanced Parameter Search Algorithms for ITL

The sections above describe some of the difficulties encountered when trying to apply standard advanced parameter search techniques to ITL problems. For each problem encountered, a viable solution or workaround has been identified allowing ITL systems to be trained with more advanced algorithms. While most of the algorithms described required some tailoring to be compatible with ITL systems, two methods, GD and RP, were sufficiently robust to be applied to ITL with no modification. RP is of

particular interest because it is also very computationally efficient. Table 5-1 summarizes the algorithms that have been applied to ITL along with the associated modifications that were required to arrive at a suitable implementation. The sections that follow will present comparisons of these methods on sample problems. Because of the complexity involved with each computation of the ITL criteria, the traditional approach of comparing algorithms by number of epochs is better performed using a lower level metric such as the number of FLOPs.

Table 5-1. Advanced parameter search algorithm modification summary for ITL.

Training Method		Modifications Required		
Acronym	Description	Adaptive Kernel Size	Performance Tracking	Relative Error Adjustment
GD	Gradient Descent			
GDA	GD w/ Adaptive Learning Rate	X	X	
GDX	GDA w/ Momentum	X	X	
RP	Resilient Backpropagation			
LM	Levenberg-Marquardt	X		
LMSC	LM – Scaled Step Size	X		X
LMLS	LM – Line Search	X		X
SCG	Scaled Conjugate Gradient	X		
CGB	CG – Powell-Beale	X		
CGF	CG – Fletcher-Reeves	X		
CGP	CG – Polak-Ribiere	X		
OSS	One Step Secant	X		
BFGS	BFGS Quasi-Newton	X		

Sample Problems

For providing comparisons, several sample problems are presented that use different ITL criterion in different manners. The first example is an unsupervised feature extraction problem that maximizes the output entropy of a feature extraction network to generate entropy-based features. The second example is a supervised training problem that maximizes ED-QMI to solve the frequency-doubling problem. Both of these

problems use a fixed kernel size. The last example attempts to solve the frequency-doubling problem with a simplified network and explores the differences between fixed and adaptive kernel sizes.

Example 1: Maximum Entropy Feature Extraction

This example is based on the maximum entropy and PCA feature extraction experiment [Pri00]. In this experiment, the input is 50 samples of a pair of two-dimensional Gaussian distributions with mean and covariance matrices,

$$\Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0.1 \end{bmatrix},\ m_1 = \begin{bmatrix} -1 & -1 \end{bmatrix},\ \Sigma_2 = \begin{bmatrix} 0.1 & 0 \\ 0 & 1 \end{bmatrix},\ m_2 = \begin{bmatrix} 1 & 1 \end{bmatrix} \tag{5-25}$$

A two input, four hidden node, single output multilayer perceptron (2-4-1 MLP) with hyperbolic tangent nonlinearities is used to extract one feature from the two dimensional data using the entropy maximization criterion. The kernel size is fixed for this example. Figure 5-5 shows typical examples of the input sample distribution with contours, initial output distribution, and final output distribution. Notice how the final trained feature contours have adapted to equally partition the input space as opposed to the untrained features. This is further illustrated by the relative uniformity of the final output distribution versus the initial distribution.

In order to compare the various parameter search algorithms, 30 simulations were conducted with each algorithm using different initial conditions and data samples. To maintain a fair comparison, the same 30 initial condition and data sample pairs are used for all of the algorithms. The learning curve for each run was saved and averaged for each algorithm. Figure 5-6 shows the average learning curves for the various training algorithms. Note that all of the advanced training techniques converge in fewer epochs than the GD methods.

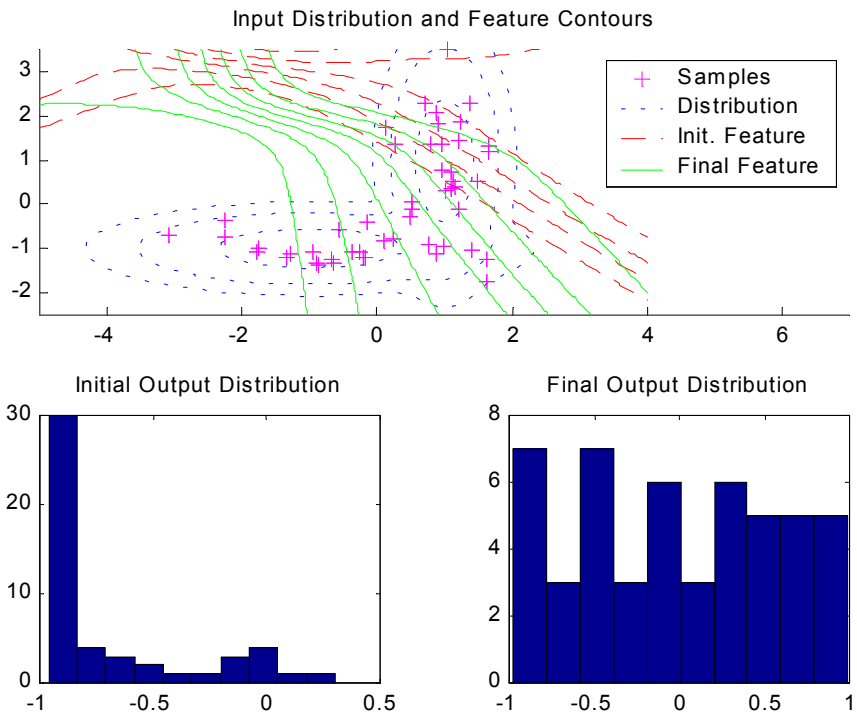

Figure 5-5. Maximum entropy feature extraction.

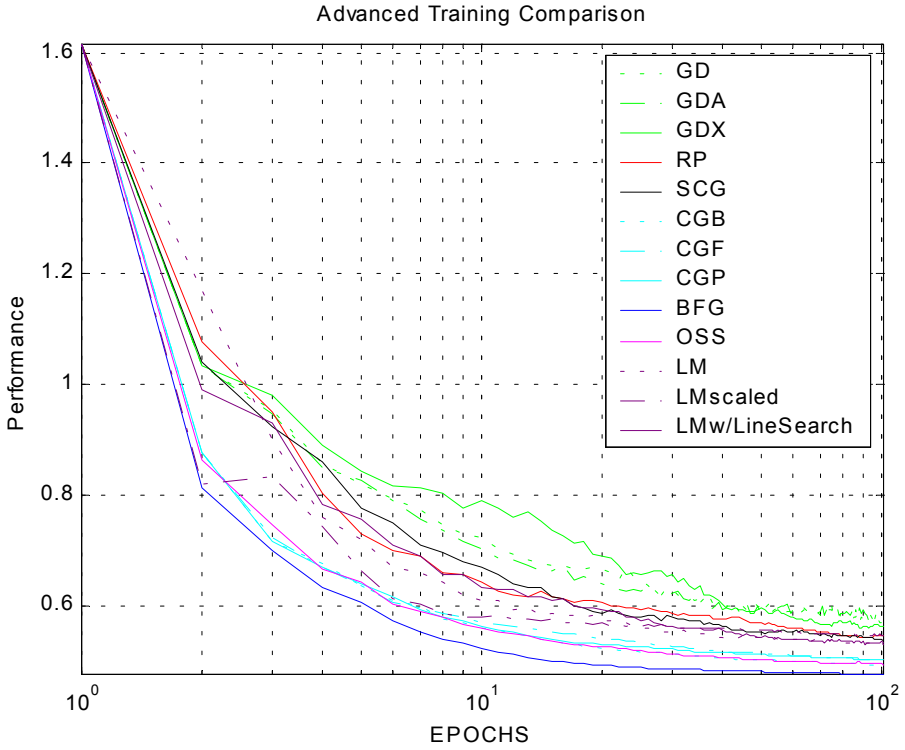

Figure 5-6. Feature extraction: Parameter search comparison in epochs.

Figure 5-7 provides a more telling measure of computational efficiency. This figure shows the average learning curve versus the number of FLOPs required to achieve the given performance level for each parameter search algorithm. In this figure, however, the computational complexity of the LMLS method makes it less efficient than GD. This is likely due to the computational expense of adding the line search to LMLS, which requires multiple ITL criterion evaluations. The GDX, LMSC, LM, and SCG algorithms are comparable in efficiency to GD while the conjugate gradient, OSS, and BFG provide superior convergence in significantly fewer FLOPs. The RP algorithm converges quickly initially, but does not achieve the terminal accuracy of some of the other advanced methods.

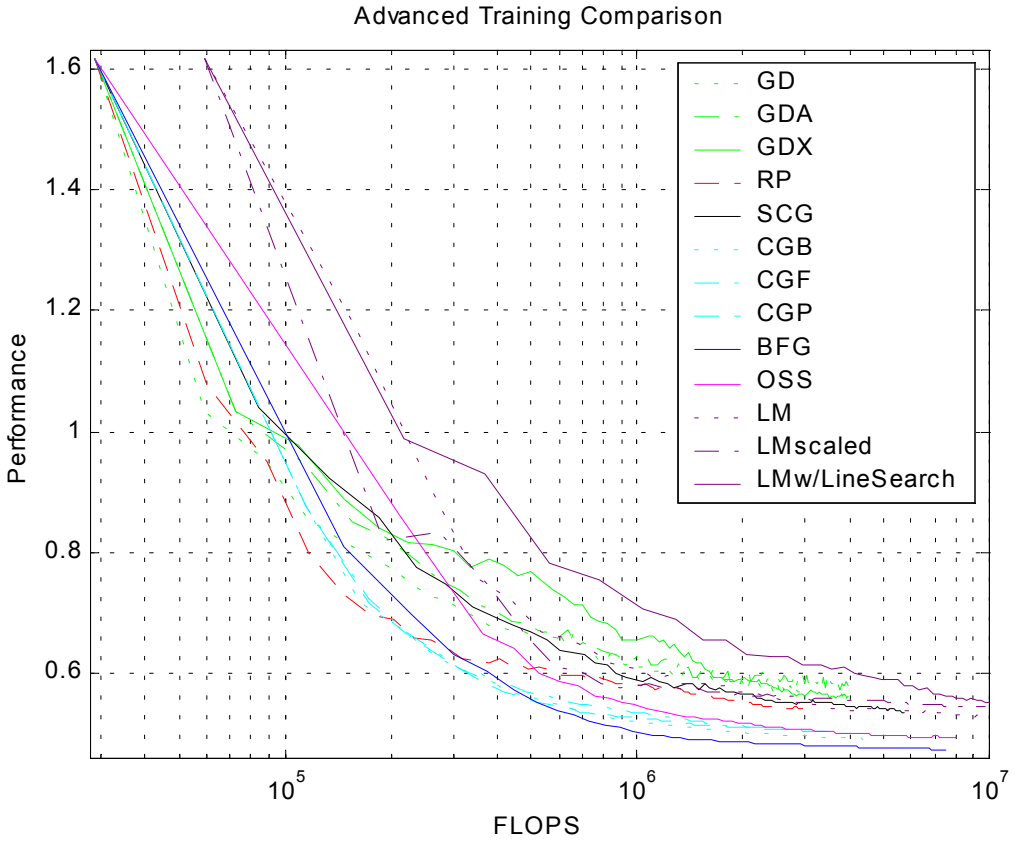

Figure 5-7. Feature extraction: Parameter search comparison in FLOPs.

Example 2: Frequency Doubler

The second example is based on the frequency-doubling problem in [Pri00]. The input signal is a sampled sine wave with a period of 40 samples. The desired output is a sine wave with twice the frequency. Five delay elements are used to create a six-input time delay neural network (TDNN). The network topology consists of two hidden units and one output unit, all with hyperbolic tangent nonlinearities. The resulting network contains a total of 17 connection weights and biases. Figure 5-8 and 5-9 show the input and desired signals and the network topology, respectfully.

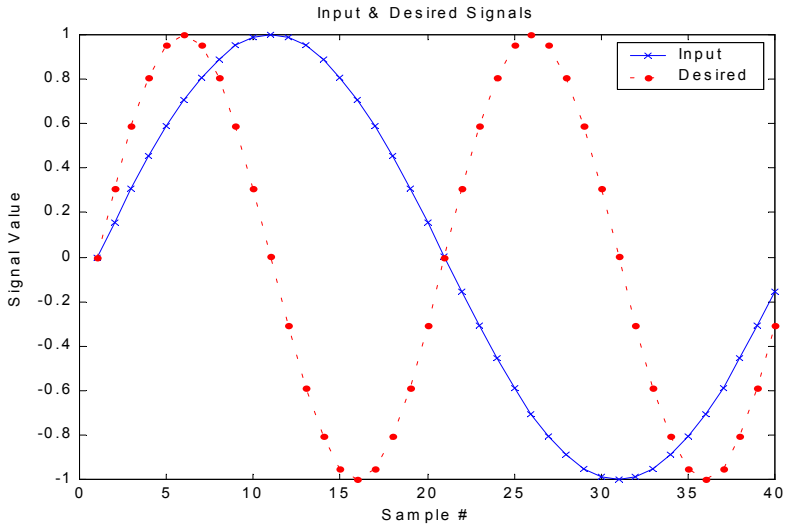

Figure 5-8. Frequency doubler: Input and desired signals

The system is trained by maximizing the ED-QMI of the network output and the desired signal with a fixed kernel size. Training was conducted using the same set of 30 different initial conditions for each of the parameter search techniques. Performance over the 30 trials is averaged for each of the advanced training techniques. At least two local minima of the performance surface were identified during the training of this system, one at -0.03 and one at -0.08, with the global minimum corresponding to a performance value of roughly -0.12. These local minima correspond to a flat signal with a single hump of

the desired sine wave and a staircase signal with level changes at each zero crossing of the desired signal as shown in Figure 5-10. Figure 5-11 plots the average of the 30 learning curves against the number of FLOPs required for the various parameter search algorithms.

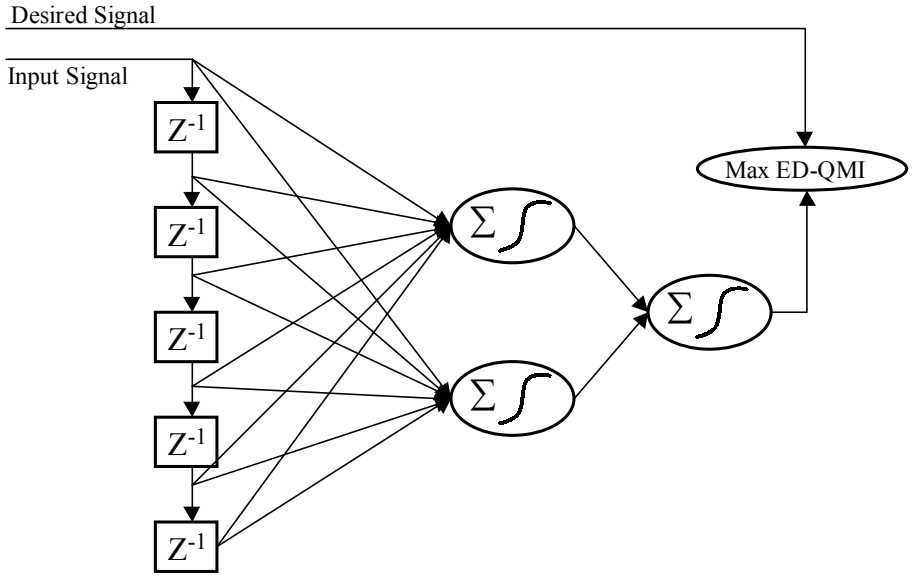

Figure 5-9. Frequency doubler: Network topology.

In this example, GD is clearly the least efficient algorithm. Some computational efficiency is achieved by each of the advanced algorithms; however, the most dramatic improvement is realized by RP and the various Levenberg-Marquardt methods. Notice that the gradient descent, conjugate gradient, and quasi-Newton methods tend to get trapped in the first local minima most frequently as shown by their high average terminal value of around -0.035 to –0.04. After this group, the LMSC, RP, LM, and LMLS methods demonstrate progressively better terminal values. Of particular interest is the efficiency of the RP method, especially in the early stages of training. While exhibiting a very good terminal average, RP has the steepest initial phase of learning. In terms of final performance, however, the LMLS method performs the best overall.

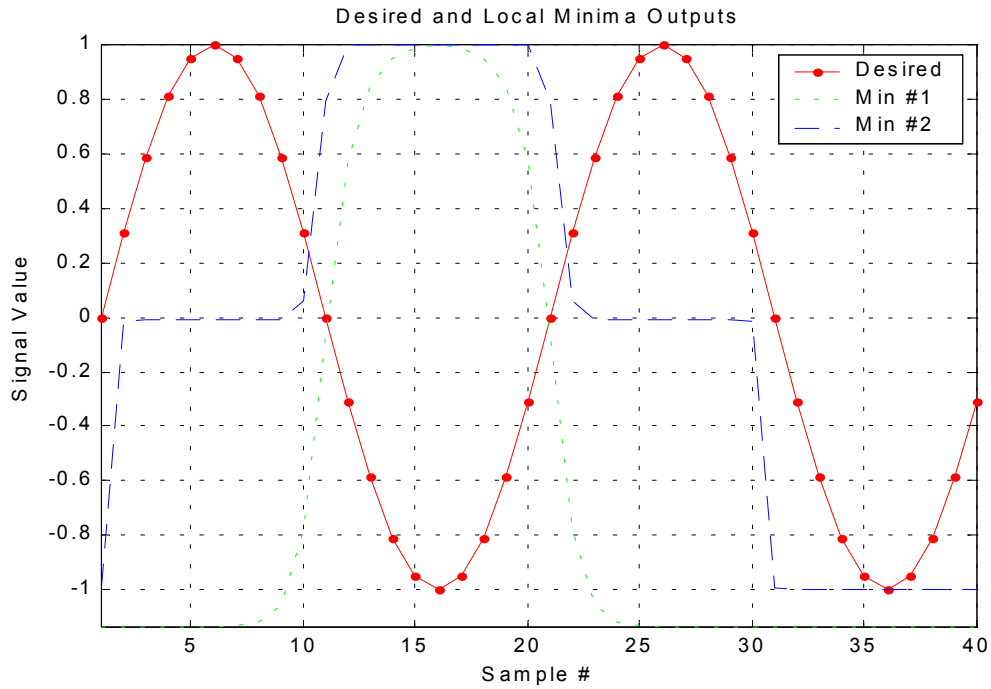

Figure 5-10. Frequency doubler: Local minima outputs.

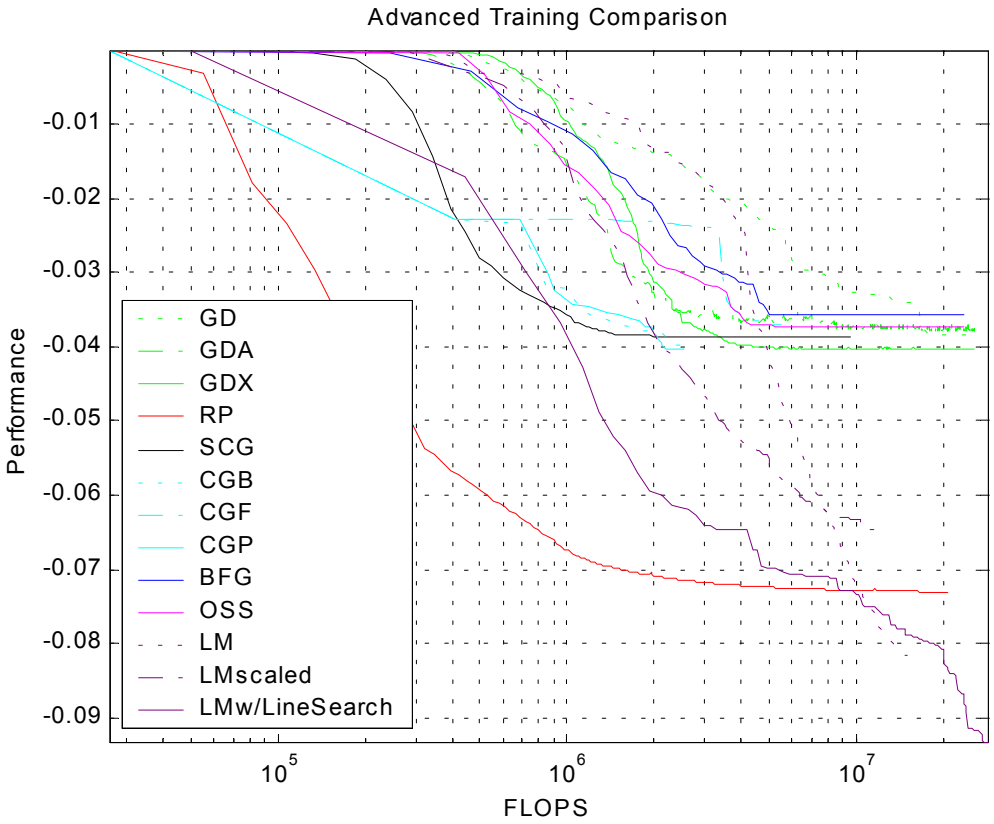

Figure 5-11. Frequency doubler: Parameter search comparison.

Example 3: Simplified Frequency Doubler

This final example, also based on the frequency-doubling problem, uses a simpler network topology to allow visualization of the effects and advantages of allowing an adaptive kernel size. Again, the input signal is a sampled sine wave with a period of 40 samples and the desired output is a sine wave with twice the frequency. In this case, a single delay element is used to create a two-input TDNN. The network topology consists of a single output unit with no bias connections and a hyperbolic tangent nonlinearity. The resulting network contains only two weights. Figure 5-8 shows the input and desired signals, however, since the network only has two weights, the network does not have the capacity to fully obtain the desired signal.

The actual performance surface for this two-parameter example is shown in Figure 5-1 for various kernel sizes. A closer examination of the performance surfaces in this figure reveals that the ED-QMI is maximized along the two lines in the weight space described by in (5-26).

$$w_2=-0.7724w_1 \quad \text{and} \quad w_2=-1.1013w_1 \tag{5-26}$$

Interestingly, these lines correspond to the linear combinations of the input sine waves that result in net phase shifts of 5, 15, 25, and 35 samples. These phase shifts have the effect of aligning the peaks and zero-crossings of the network output with the peaks of the desired signal. The optimum outputs for this network using ED-QMI based on the weight space equations above are given by the equations in (5-27), of which the first two are shown graphically in Figure 5-12 for $C_1=C_2=1$. Note that if C_1 and/or C_2 are negative, the 5 and 15 phase shifts become 25 and 35, respectively. Thus, the four global minima for this performance surface correspond to the blue and cyan signals in Figure 5-12 and their respective negatives, subject to scaling.

$$\begin{aligned} y_1 &= C_1 \sin 2\pi f (n-5) \\ y_2 &= C_2 \sin 2\pi f (n-15) \\ y_3 &= C_3 \sin 2\pi f (n-25) \\ y_4 &= C_4 \sin 2\pi f (n-35) \end{aligned} \tag{5-27}$$

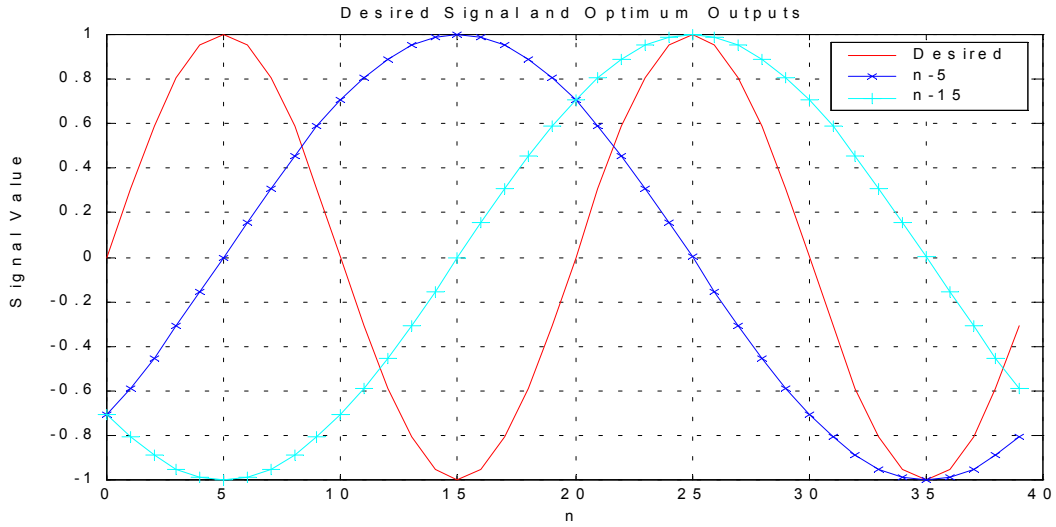

Figure 5-12. Desired and ED-QMI optimal signals.

The nonlinearity in the network topology limits the magnitude of the constants C_i to avoid distortion caused by saturation of the outputs. The four global minima for the kernel size of $\sigma^2 = 0.01$ are approximately (-/+2.2, +/-2.85) and (-/+4.35, +/-3.95), which are indeed co-linear with the optimum described above. Performance also deteriorates near the origin as might be expected. In contrast, when using the MSE criterion with this reduced size network, optimization drives the two weight values to zero for a net zero output. Note that even in this simple example, the ITL method tries to pass as much information as possible through the limited topology network unlike the MSE.

Experiments were conducted using three sets of initial conditions: IC#1 (0,1) near the first minimum, IC#2 (-4,3) near the second minimum, and IC#3 (4,0) near a plateau. As a baseline, the network is trained using a fixed kernel size of $\sigma^2 = 0.01$. Results show, as expected, that the training function's convergence is dependent on the initial

conditions (IC). Figure 5-13 shows the weight tracks for the RP, GDA, and LMSC algorithms for the three different initial conditions. The circles indicate the final values. All three IC's yield different convergence. It is evident from the weight tracks that the LMSC algorithm performs very well in reaching the local minimum that is closest to the IC with fixed kernel size. Also, note that for IC#3 all three algorithms fail to find a global minimum, but rather, get stuck in a shallow minimum in the plateau.

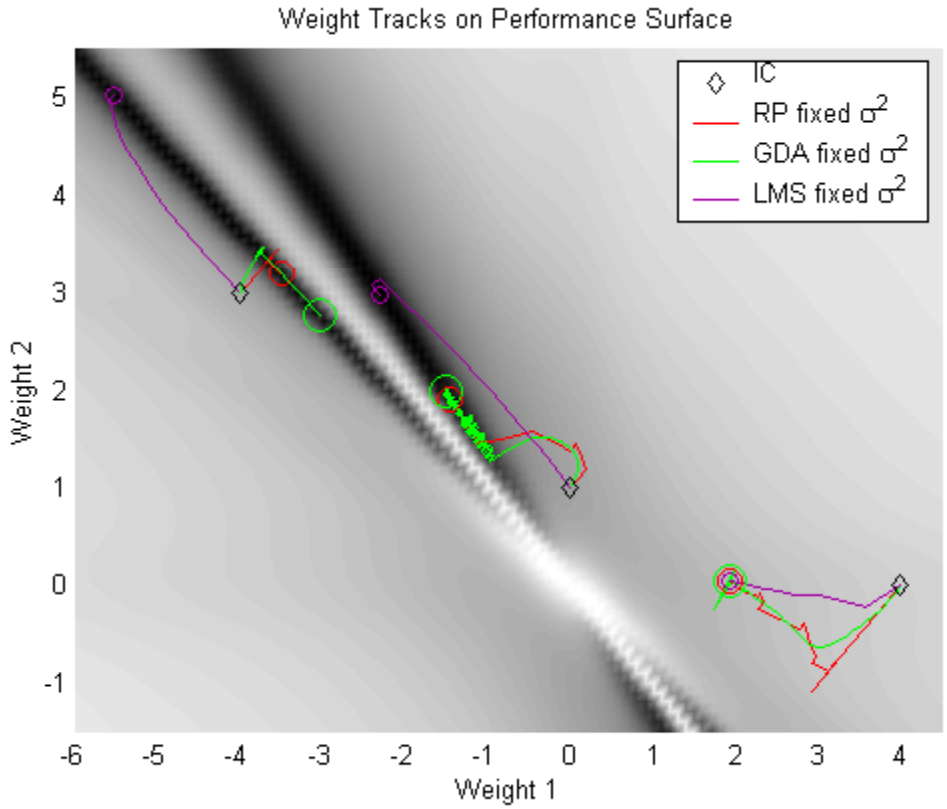

Figure 5-13. Fixed kernel size weight tracks.

Next, the same experiment was conducted using an adaptive kernel size. As noted previously, the selection of an appropriate kernel size for ITL is crucial for proper learning to occur. Because of the difficulty associated with selecting a single fixed value of kernel size, an adaptive scheme for determining kernel size based on the distribution of the data is desirable. Some schemes involve computing the kernel size to ensure that a

number of interactions (from 1 to N) occur between outputs. In this particular example, a 1-N mean scheme that averages the kernel size for one interaction with that for N interactions as described in (5-28) is used, although other schemes could be used as well.

$$\sigma = \frac{\min(nearestneighbordist(\mathbf{Y})) + \max(nearestneighbordist(\mathbf{Y}))}{2} \tag{5-28}$$

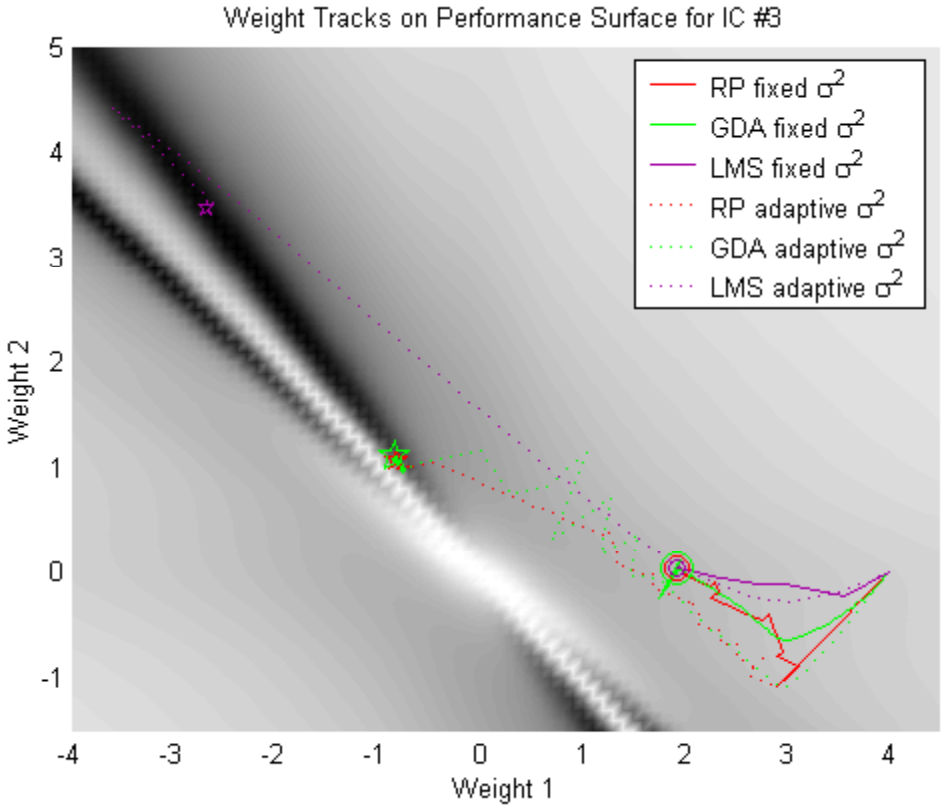

Figure 5-14. Fixed and adaptive kernel size weight tracks for IC #3.

The weight track results for this 1-N mean adaptive kernel size method are shown in Figure 5-14 for IC#3. The stars indicate the final values for the adaptive case. Note that the algorithms were able to converge towards one of the global minima in all three cases with an adaptive kernel size. The methods were able to move past the plateau that hindered algorithms in the fixed kernel size case. This confirms that there is indeed a benefit to using an adaptive kernel size. Figure 5-15 plots both the kernel size and network performance for the RP method. Note that the kernel size is relatively large until

the training performance begins to improve as it goes past the plateau. Then, both the kernel size and the performance decrease sharply.

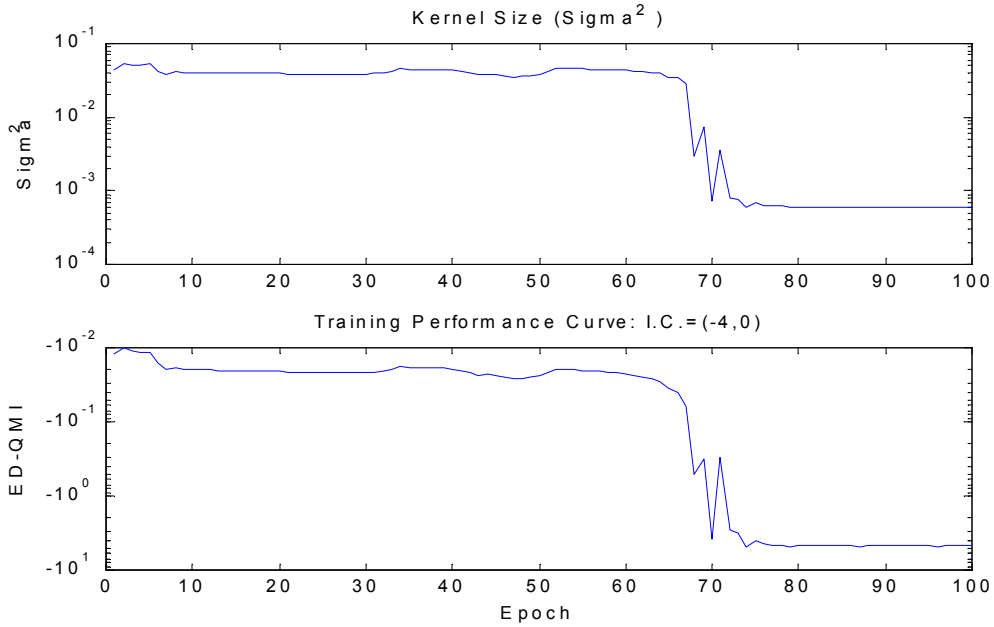

Figure 5-15. Kernel size and training performance for RP and IC#3.

Conclusions

Experimental results have demonstrated that the application of properly tailored advanced parameter search algorithms can provide significant computational savings for the training of ITL systems over standard gradient descent. In the examples provided, computational savings greater than two orders of magnitude were realized in some cases. As with the training of traditional systems, the training of different ITL systems is not always accomplished most efficiently by one single algorithm. Like their standard counterparts, the ITL advanced search methods have similar advantages and disadvantages in terms of storage, complexity, and robustness when applied to problems with different number of samples, weights, and criterion. In the examples presented, the LM methods and the RP method were overall the most efficient. However, the results

suggest that in general, the application of several algorithms might be necessary to avoid the trappings of a single algorithm applied to a specific problem.

The results have also demonstrated the importance of the kernel size to the training of ITL systems. The coupling of an inappropriate kernel size with certain initial conditions was shown to render the search algorithms susceptible to shallow local minima in the performance surface that were merely artifacts of the impact of too small a kernel size on PDF estimation. A 1-*N* mean kernel size adaptation algorithm was described that ensures some level of interaction between the system outputs. The benefits of using an adaptive kernel size during training, and the need to tailor the parameter search algorithms to accommodate this adaptation were also shown.

CHAPTER 6
BATCH TRAINING APPROACH

Introduction

The algorithm optimization of ITL criteria computation and the application of advanced parameter search techniques for ITL training have resulted in significant reductions in the computational demands of training ITL systems. However, despite these improvements, the computational complexity of a given ITL problem remains proportional to the number of samples, N^2. Because of this fact, the application of ITL to problems with very large data sets or continuously increasing data length, such as continuous time series data, continues to present a computational challenge for practical implementations. In this chapter, a batch training method for ITL systems is presented that segments the original data into batches in order to reduce the N^2 computational dependency. The stochastic information gradient (SIG) method of Erdogmus [Erd01], which was developed concurrently with and independently of the batch method presented herein, is shown to be a special case of the batch training approach. Several examples are used to show the benefits of batch training for ITL applications.

Batch Training Approach

Overview

The investigation of batch training for ITL is motivated by the increasingly slow training that results from the combination of large data sets with the N^2 computational complexity of ITL criteria. Fundamentally, the batch approach uses subsets of the entire data set to train the learning system incrementally with smaller batches of appropriately

sampled data. Theoretically, the justification for being able to conduct ITL training using batch techniques depends on the validity at least one of two assumptions. The first is that learning can be accomplished incrementally with subsets of the full dataset. The second is that there is some level of redundancy in the data. It has been empirically demonstrated for many problems that the first assumption holds true and computationally efficient training can be accomplished using the SIG method [Erd01]. The utility of the SIG method shows that sufficient information is contained in sequential samples of a properly sampled time series to incrementally learn the characteristics of a signal.

Consider, however, the conditions that might cause the SIG method difficulties. Suppose that a noisy time series is sampled at a very high rate relative to its bandwidth. In this case, the differences between adjacent samples corrupted with noise might not be sufficient to efficiently use the SIG compared to a full ITL implementation. However, since the system has been oversampled, the full data set would necessarily contain redundant information. In this case, simply resampling the full data set at a lower sample rate and applying ITL to the reduced size set could reduce computational complexity considerably.

In general, if a signal space is oversampled, discarding redundant information can provide computational savings. If a signal space is appropriately sampled, the results herein along with those for SIG research suggest that incremental learning with computational savings can be achieved. Since it is often unclear whether the density of the sampled data is too sparse or too tight, the primary difficulty lies in the selection of the most efficient batch training implementation. Some guidelines for the selection of batch training parameters are presented later in this chapter.

Batch Training Parameters

The implementation of the concept of batch training introduces several new design parameters in addition to the typical training parameters that must be selected using standard ITL techniques. Among these are the batch sample size, the number of epochs per batch, the number of batches, and the batch selection method. The description and selection of these parameters is discussed in detail below along with other considerations that are important to a proper batch ITL implementation.

The batch sample size parameter, *Bsize*, represents the size of the batch or the number of samples to be used for the batch ITL computation. The relationship between the batch size and the original number of samples, *N*, governs the maximum computational reduction achievable using a batch implementation. The number of epochs per batch, *Neb*, is the numbers of iterations given to a particular parameter search algorithm to train with each batch. The number of batches, *Nb*, represents the number of times a new batch is generated and used for training. A simple example is presented below to provide a clear illustration of these new parameters and the potential advantage of using a batch training technique.

This example is based on a time-series prediction problem, which is the second example in [Erd01]. In this problem, the input data set consists of N=32 samples of a composite sinusoidal signal generated by

$$x(t) = \sin 20t + 2\sin 40t + 3\sin 60t \tag{6-1}$$

and is sampled at 100 hertz. Each of these 32 samples is a two dimensional vector consisting of sequential signal samples to be used to predict the next sample. Thus, the full input to the ITL system can be described by (6-2).

$$\mathbf{X}_{[2x32]} = \left[\begin{bmatrix} x(1) \\ x(0) \end{bmatrix} \quad \cdots \quad \begin{bmatrix} x(32) \\ x(31) \end{bmatrix}\right] = \left[\mathbf{x}(1)_{[2x1]} \ldots \mathbf{x}(32)_{[2x1]}\right] \tag{6-2}$$

For illustration purposes, suppose that when using the full input set for ITL training, satisfactory performance is obtained after *Ne*=100 epochs. Note that this corresponds to a batch implementation where *Bsize*=*N*, *Nb*=1, *Neb*=*Ne*, therefore the single batch is the entire input set. The computational complexity of this approach is proportional to the product of the square of the number of samples with the number of epochs (i.e., $N^2 \times Ne$=102400) or equivalently, $Bsize^2 \times Nb \times Neb$.

Now consider a batch implementation where the batch size is one quarter of the entire data set (i.e., *Bsize*=*N*÷4=8). The input subset batches, based on a sequential nonoverlapping selection method, would be as follows:

$$\begin{aligned} \mathbf{X}_{B1[2x8]} &= \left[\mathbf{x}(1)_{[2x1]} \ldots \mathbf{x}(8)_{[2x1]}\right] \\ \mathbf{X}_{B2[2x8]} &= \left[\mathbf{x}(9)_{[2x1]} \ldots \mathbf{x}(16)_{[2x1]}\right] \\ \mathbf{X}_{B3[2x8]} &= \left[\mathbf{x}(17)_{[2x1]} \ldots \mathbf{x}(24)_{[2x1]}\right] \\ \mathbf{X}_{B4[2x8]} &= \left[\mathbf{x}(25)_{[2x1]} \ldots \mathbf{x}(32)_{[2x1]}\right] \end{aligned} \tag{6-3}$$

thus producing four batches or *Nb*=4. Using each of these four batches, the learning system would be trained for *Neb* epochs with the weights remembered from batch to batch. For comparison purposes, suppose that the case where *Nb*=4 and *Neb*=100 produces similar performance to the full ITL training case. In this case, the computational complexity is proportional to $Bsize^2 \times Nb \times Neb$=25600, a reduction by a factor of four. The relationship between the computational demands of batch training and full ITL training is summarized by (6-4).

$$\frac{Bsize^2 \times Nb \times Neb}{N^2 \times Ne} \tag{6-4}$$

From (6-4), it is clear that linear reductions in the *Bsize* with less than quadratic increases in $Nb \times Neb$ will result in more computationally efficient training. This potential for computational savings was the motivating factor behind the development of a batch training approach for ITL.

Batch Selection Methods

Various approaches exist for selecting the batch subsets from the entire data set. The SIG method simply selects sets of sequential data samples. Random sampling is also a viable approach. Another approach is to attempt to select a representative subset of the entire data set based on the distribution of the original data. Although a true implementation of this method can be computationally expensive, simplified alternatives exist. The following three batch selection methods have been implemented for ITL batch training to accommodate any *Bsize* from two to *N*:

1. Sequential: Selects sets of sequential samples based on the index value.
2. Random: Selects a random subset of unique samples with uniform likelihood over the sample index.
3. Representative: Sorts the original sample set based on sample vector magnitude and then selects uniformly over the sorted distribution.

Comparisons of the performance of these methods are presented in the sections below.

Other Batch Training Considerations

Other critical decisions in the design of a batch training approach include the selection of an adaptation method along with its associated training parameters. Experimental results have demonstrated that the ideal set of batch training parameters for one training method is not necessarily optimal for a different training method. For example, the ideal batch size for GD learning might be different from that for LM learning. The results presented in the sections below try to capture some of the

characteristics and tradeoffs between various adaptation methods and batch training parameters.

Stochastic Information Gradient (SIG) Method

The SIG method was developed to provide an online adaptation algorithm for ITL system training with MinErrEnt criterion. The method is derived from an approximation to the gradient of the error entropy cost function that yields a stochastic instantaneous estimation for the gradient [Erd01]. The approach estimates the gradient based on a window of the most recent *Nsig* samples of a time series. The special case where *Nsig*=2 represents the standard SIG implementation. Interestingly, this approach is a special case of the batch training approach described above where *Bsize*=*Nsig*=2 and *Neb*=1 with sequential batch selection, gradient descent adaptation, and error entropy criterion.

Batch Training Comparison Results

Performance Comparison Metric

The ideal method for the comparison of different batch training implementations plots the system performance, as measured by the value of the information-based criterion for the problem at hand, versus the computational complexity of the implementation. The optimal implementation would be the one that achieves the desired performance level with the least computational complexity. During the actual batch training process, however, the learning system adapts its parameters based on an estimate of the system performance using a subset of the input space. Because of this, the runtime performance estimates are noisy and not a good metric for fair comparison of the different implementations. To account for this, the learning system weights are saved for each epoch during batch training. After training, these stored weights are used to compute the actual performance using the full input data set for each epoch. The number

of FLOPs is used to measure the computational complexity of the algorithms for each epoch during run time.

Variations of the SIG

Figure 6-1 compares several variations of the standard SIG method with the time-series prediction problem described above. In these comparisons, GD is the parameter adaptation method with a step size of 0.2. Recall that the standard SIG method corresponds to *Bsize*=2, *Neb*=1, and sequential batch selection. In Figure 6-1, all combinations of *Bsize*=[2,3,4] and *Neb*=[1,2,3] are shown. Note that early in training, increasing the batch size alone has an adverse effect on training efficiency. However, in the later stages of training, the batch sizes of three and four converge more rapidly than the batch size of two as denoted by the steeper slopes of these curves. This suggests that more efficient training might be achieved by gradually increasing the batch size as training progresses. Note also that increasing the epochs per batch tends to have a positive effect on efficiency. Clearly, several implementations converge faster than the standard two-sample SIG method. Also of interest is that the performance of the full *N*-sample ITL implementation requires approximately12MFLOPs and is still far from convergence on this scale. This clearly shows the computational savings that can be realized using batch methods.

Batch Training with Advanced Parameter Adaptation

This section compares several variations of the batch training method when combined with some of the advanced parameter search techniques from Chapter 5. Again, the problem is the time-series prediction problem described previously. In these comparisons, five sets of batch parameters are used as described in Table 6-1. Note that the first case corresponds to a full ITL implementation.

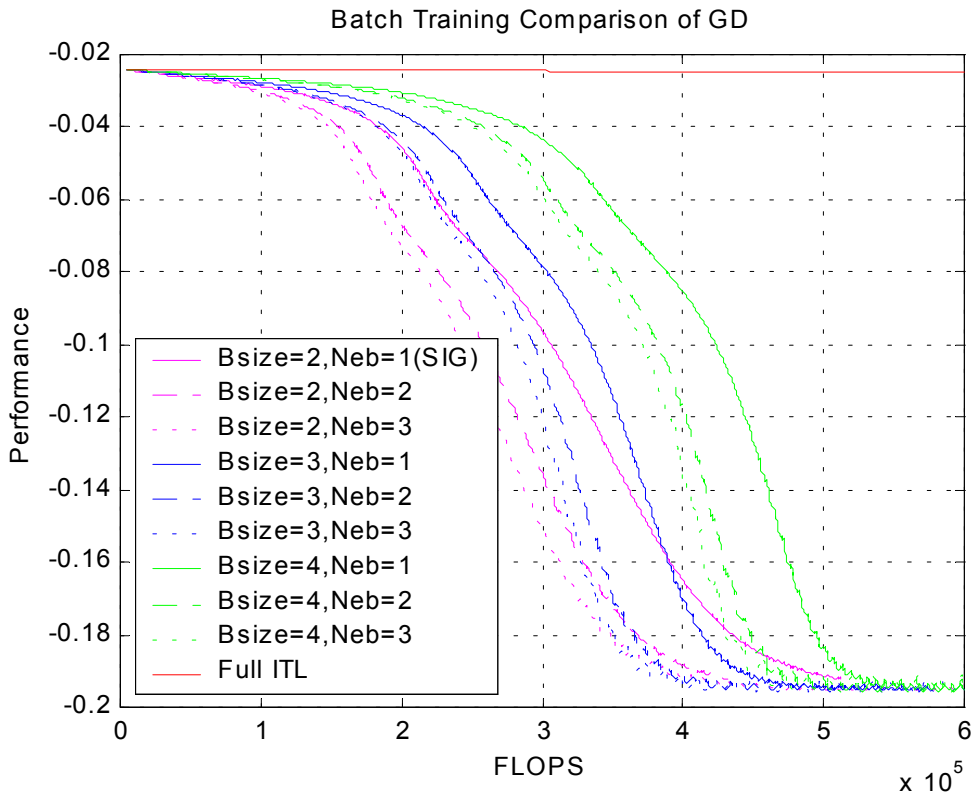

Figure 6-1. Batch training comparison for GD.

Table 6-1. Batch training parameter settings.

Setting #	Bsize	Neb	Nb
1	32	100	1
2	16	50	4
3	8	25	16
4	4	12	64
5	2	6	256

Each setting is applied and the system is trained using the SCG, RP, and LM methods for ITL training. Figures 6-2, 6-3, and 6-4 show the training performance comparisons for SCG, RP, and LM respectively using sequential batch selection. For the SCG case in Figure 6-2, note that the convergence gets gradually smoother as the batch size is increased. This is the general trend for all adaptation methods. However, the computational efficiency reaches an optimum around *Bsize*=4 and *Bsize*=8 for both SCG and RP. Notice that in each case, the optimum performance is neither the full input set

where *Bsize*=32 nor Setting #5 where *Bsize*=2. For comparison, the performance of the standard SIG method is show on each of the figures. Notice that computational savings of greater than an order of magnitude are achieved in many cases by combining advanced parameter search with batch methods.

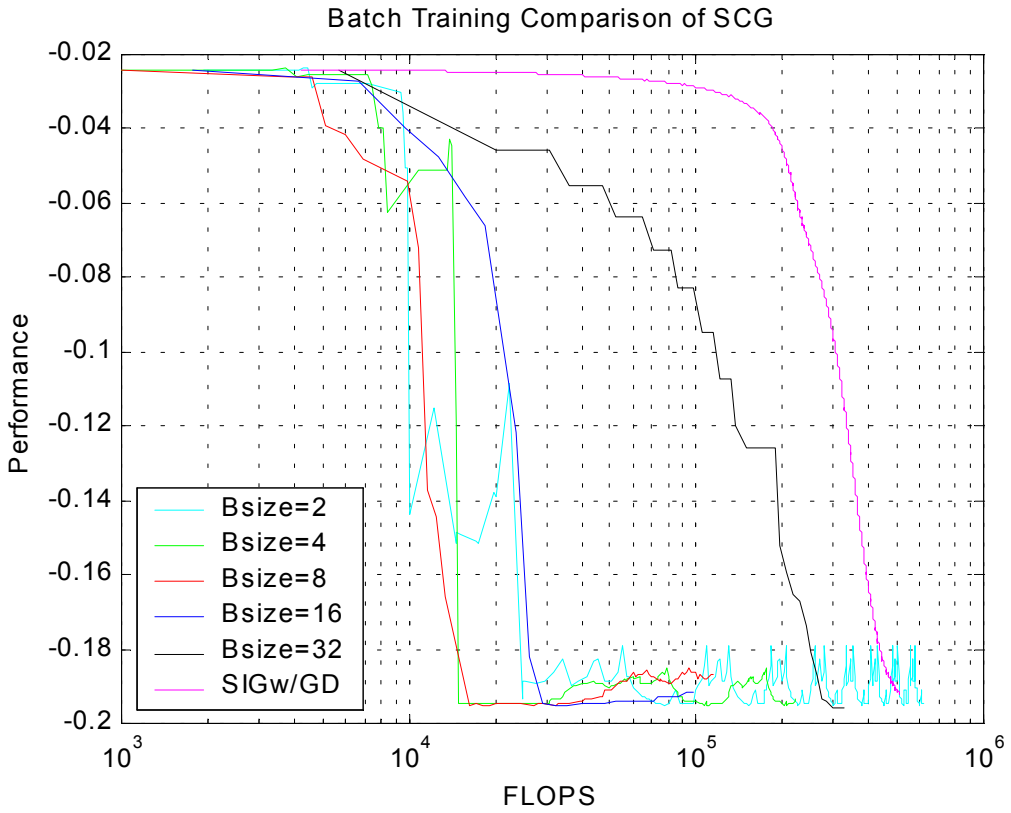

Figure 6-2. Batch training comparison for SCG.

Batch Selection Methods

Figure 6-5 compares the three proposed batch selection methods for the time series prediction problem described previously. For each case, *Bsize*=3 and *Neb*=3 with GD parameter adaptation. Notice that in these cases, sequential selection performs the best while representative selection is worst. This is understandable since there is no noise in the system and the representative case has a significant amount of overhead in computing the vector magnitudes.

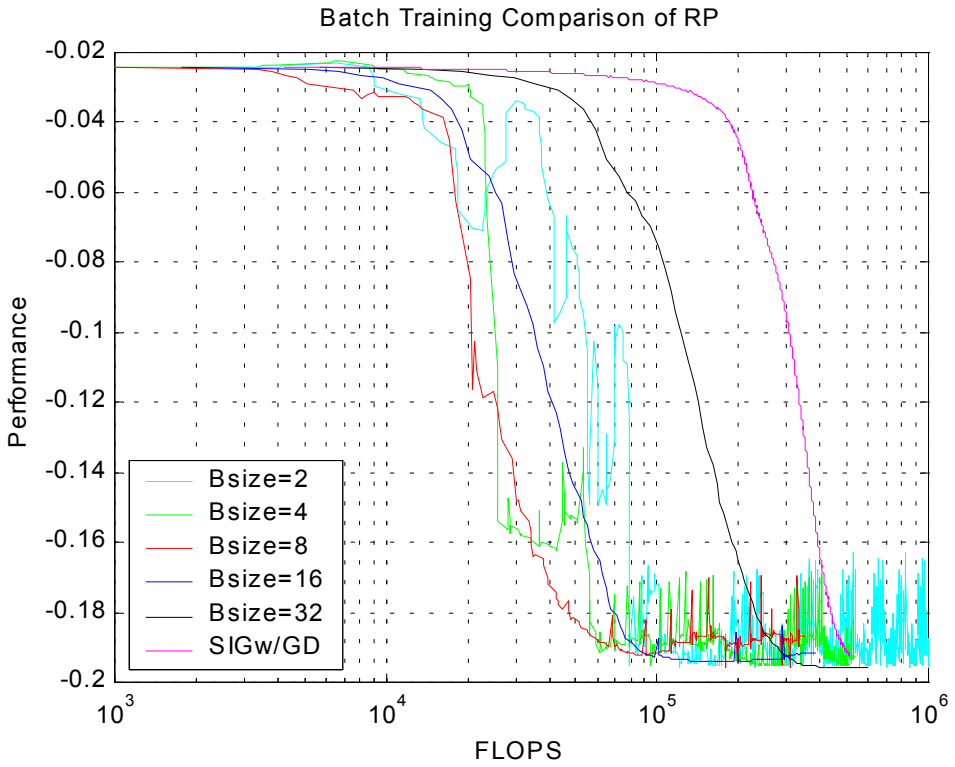

Figure 6-3. Batch training comparison for RP.

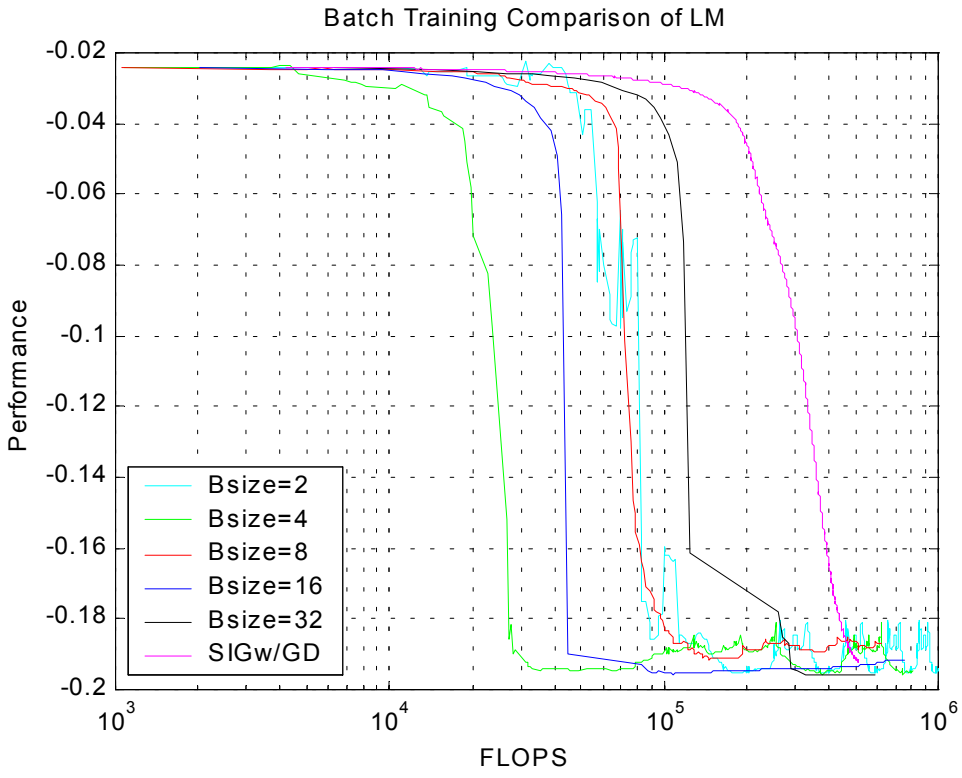

Figure 6-4. Batch training comparison for LM.

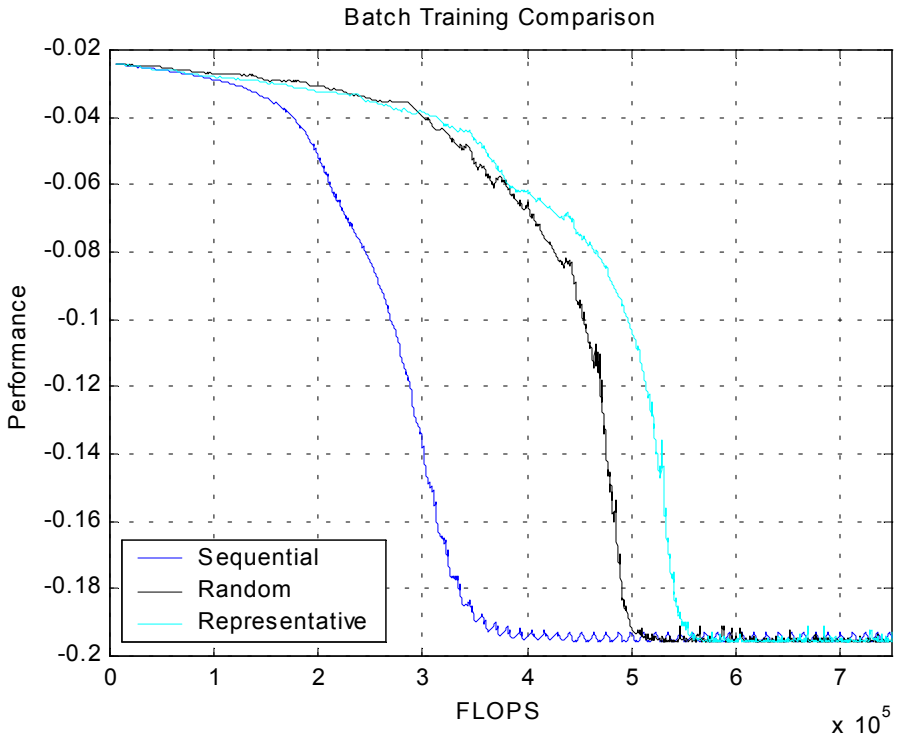

Figure 6-5. Batch training selection method comparison with GD.

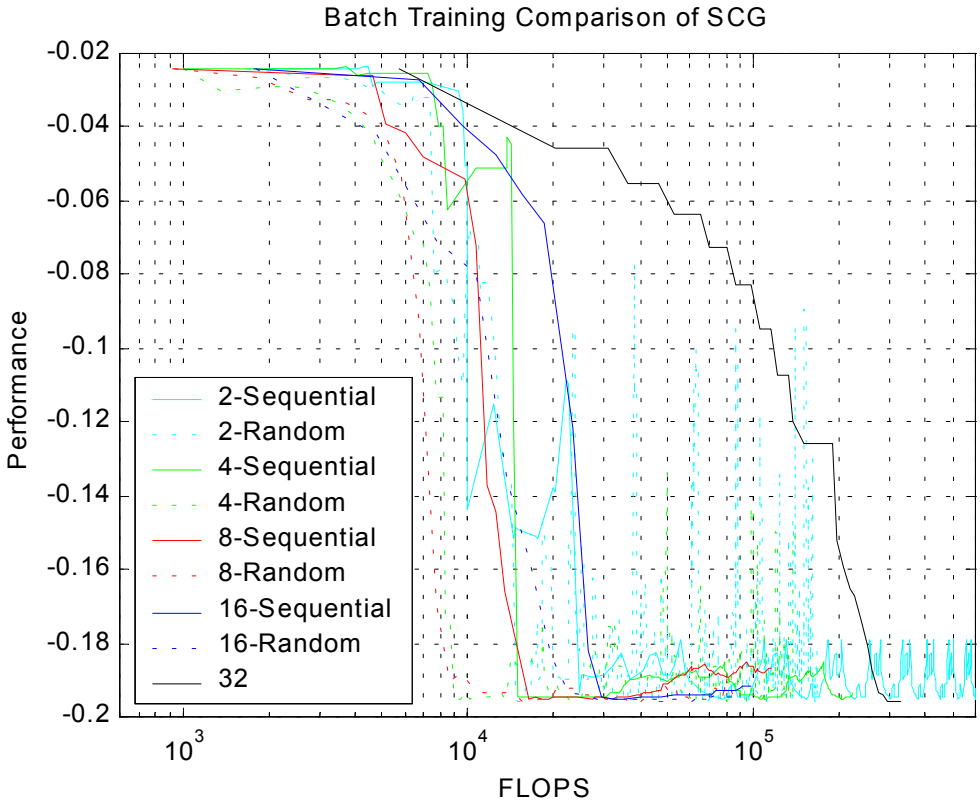

Figure 6-6. Batch training selection method comparison with SCG.

When advanced parameter adaptation algorithms are incorporated, the optimum batch selection technique is less clear. Figure 6-6 shows the performance of SCG with both sequential and random batch selection for the five batch training parameter settings described in the previous section. Notice that in all of the cases except for *Bsize*=2, performance is improved using random versus sequential batch selection.

Batch Training Comparison with Additive Noise

In this example, a random noise signal, $n(t)$, is added to the time-series prediction problem from above to further investigate the effects of the batch training parameters on the performance of ITL. For this example, a zero-mean normally distributed noise source with σ =0.5 is added to the system described in (6-1) to yield the system described by,

$$x(t) = \sin 20t + 2\sin 40t + 3\sin 60t + n(t) \tag{6-5}$$

and is again sampled at 100 hertz. Figure 6-7 shows the original signal along with the noise-corrupted signal described by (6-5).

Figure 6-8 shows all combinations of *Bsize*=[2,3,4] and *Neb*=[1,2,3]. Once again, in early training, the increased batch size has an adverse effect on training efficiency. In the later stages of training, the large batch sizes not only converge more rapidly but also attain a better level of performance than the cases where *Bsize*=2.

Figure 6-9 compares the three batch selection methods for the noisy time series prediction problem. Again, for each case, *Bsize*=3 and *Neb*=3 with GD parameter adaptation. Notice in this case that although the sequential selection performs the best initially, the random and representative batch selection methods converge faster in the latter stages and achieve a more optimal result. These results confirm the importance of proper batch selection and batch size determination.

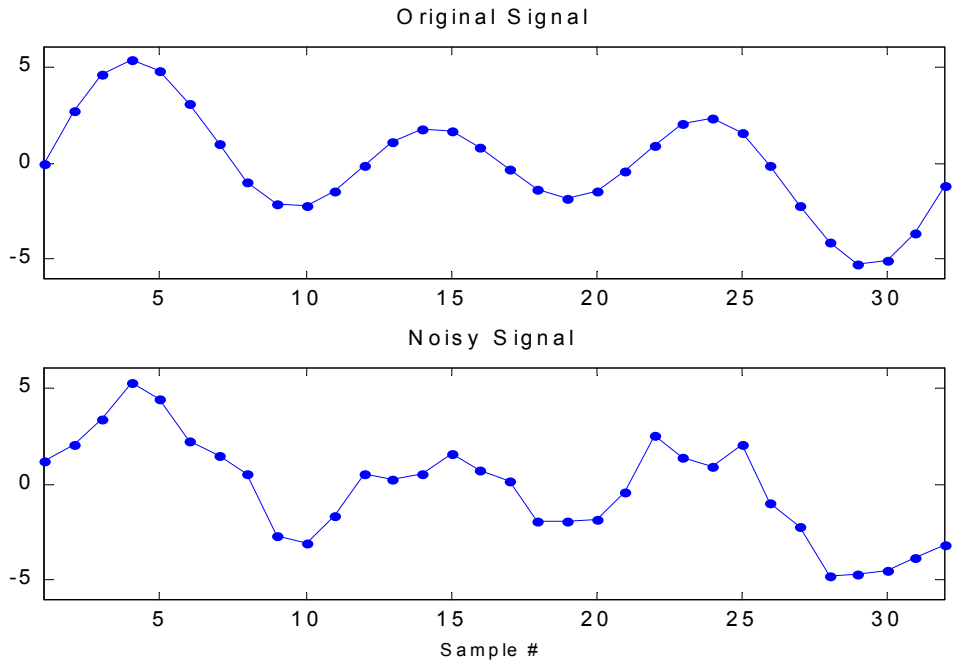

Figure 6-7. Original and noise corrupted signals.

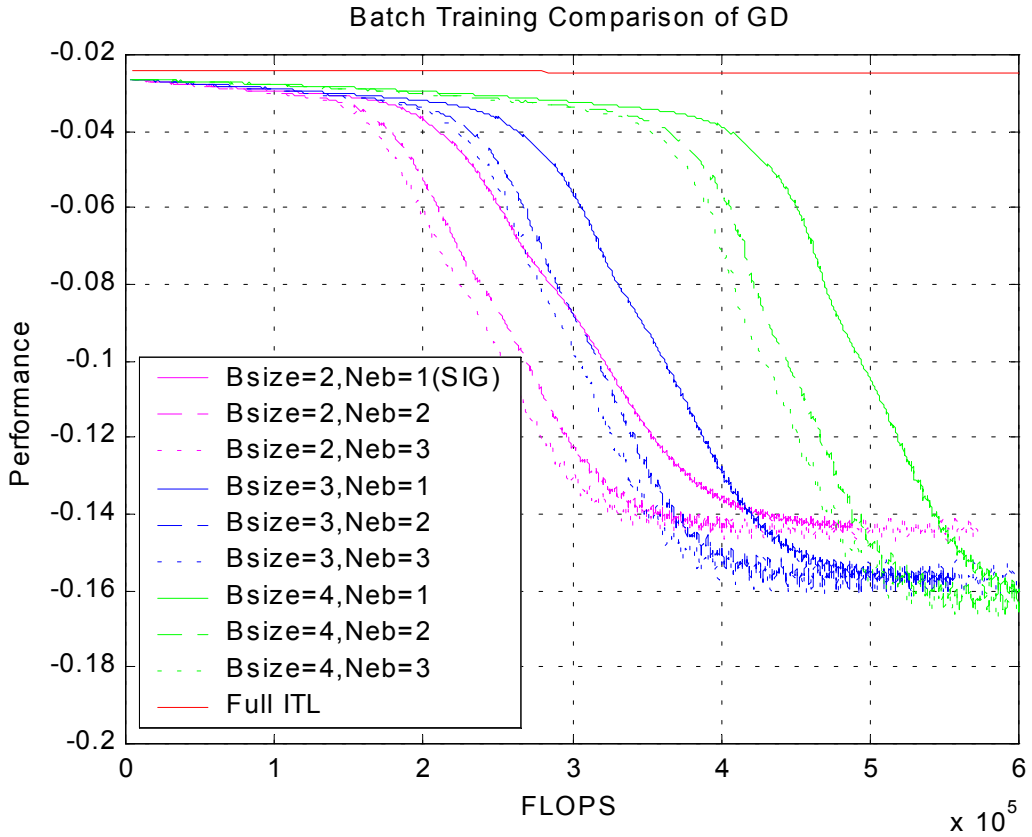

Figure 6-8. Batch training comparison: GD with noise.

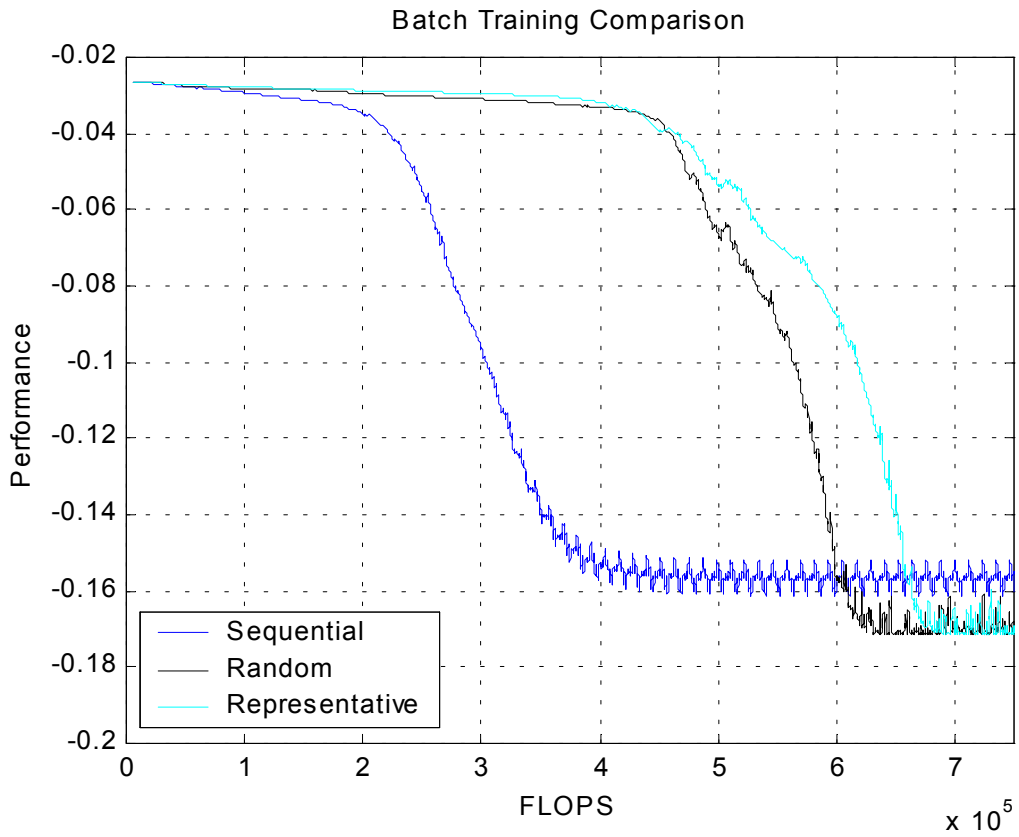

Figure 6-9. Batch training selection method comparison: GD and noise.

Batch Training Comparison with Oversampling

In this example, the time-series prediction problem from above is oversampled to further illustrate some of the effects of the batch training parameters on the performance of ITL. Equation (6-1) is now sampled at 1000 hertz rather than 100 hertz to yield a data set that contains 320 samples. Figure 6-10 shows all combinations of *Bsize*=[2,3,4] and *Neb*=[1,2,3] with sequential batch selection and GD adaptation. Note that the increased batch size has a positive effect on efficiency throughout the training process. The larger batch sizes now converge with approximately twice the efficiency of the cases where *Bsize*=2. Also, note that once again the increases in *Neb* also result in improved efficiency for all batch sizes.

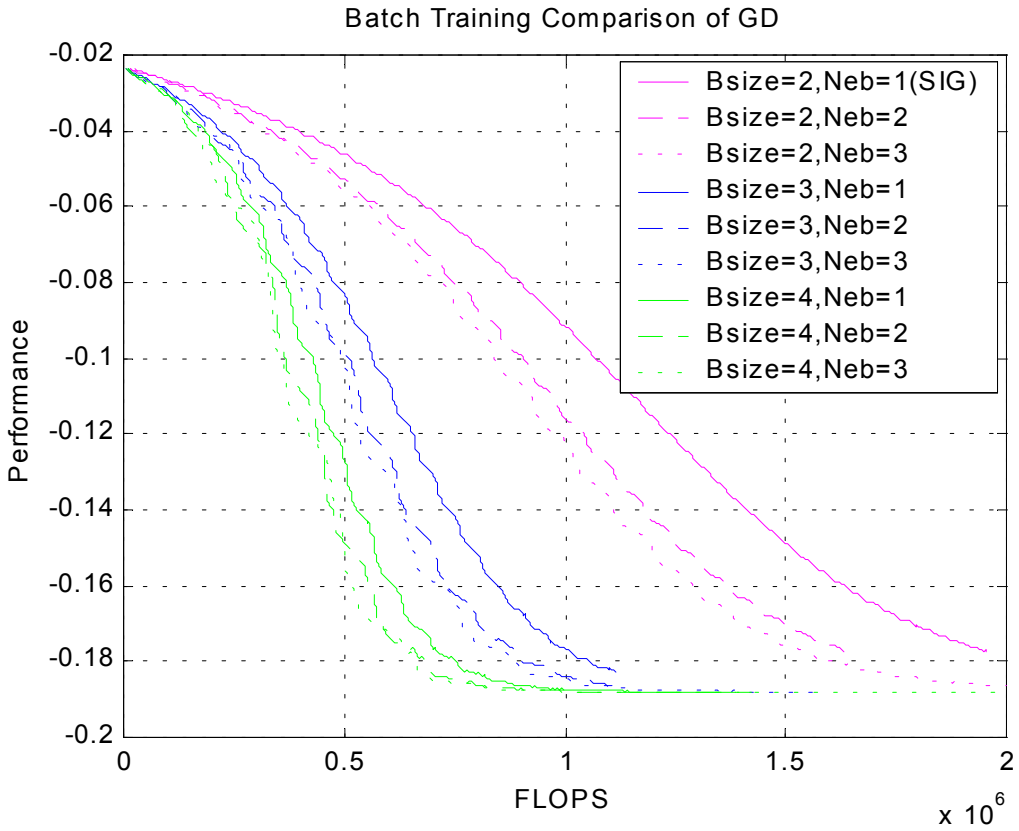

Figure 6-10. Batch training comparison: GD with oversampled.

Figure 6-11 compares the three batch selection methods for the oversampled time series prediction problem. For each case, *Bsize*=3 and *Neb*=3 with GD parameter adaptation. Notice that in this case the random selection performs the best, followed by the representative and finally the sequential. This is as expected since the oversampling results in less information being carried in sequential samples than in the baseline sampled case. Also, note that the random sampling allows the system to achieve a more optimal level of performance. This is because random sampling can explore the full bandwidth of sample data while the spectral window produced by sequential sampling yields one with a fixed bandwidth. These results further illustrate the importance of applying the proper batch selection method and batch size based on the problem at hand.

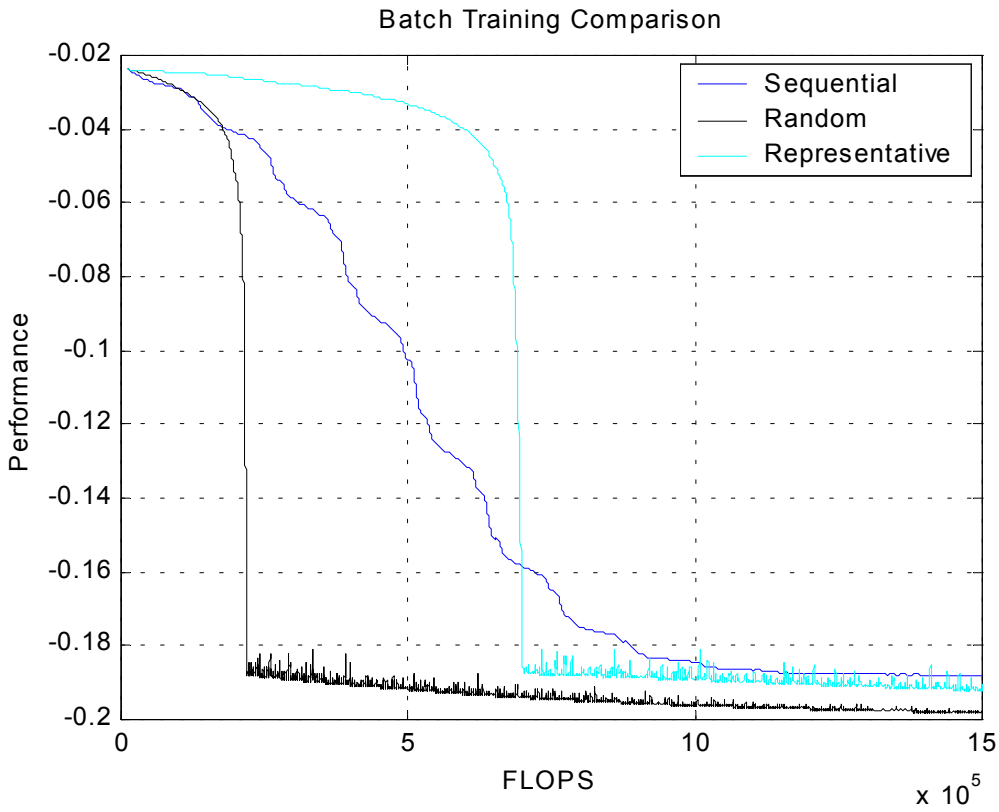

Figure 6-11. Batch training selection method comparison with GD: Oversampled.

Discussion on Batch Parameter Selection

Results have shown that increasing the number of epochs per batch, *Neb*, contributes to improved efficiency. This phenomenon is perhaps most relevant to GD adaptation since the step size selection is typically conservative and inherently suboptimal. Because of this, efficiencies are gained by performing several steps of adaptation with a given batch when compared with the overhead of selecting and beginning training with a new batch. The tradeoff for increasing this parameter is increased magnitude of oscillations during convergence as shown in Figures 6-1, 6-8, and 6-10. Larger selections for *Neb* result in larger oscillations and reduced overall efficiency. A selection of two to three epochs per batch has consistently produced good results.

Selection of the optimum batch size, *Bsize*, warrants more careful considerations based on the characteristics of the problem to be addressed. Results have demonstrated that a batch size of two provides very efficient initial convergence for appropriately sampled data even in the presence of noise. Larger batch sizes, however, performed more effectively in terminal training for most cases particularly when noise is added to the system. In the case of oversampled data, larger batch sizes provided a substantial benefit throughout. However, since it is often impossible to determine how appropriate the sample rate is for the underlying characteristics of the signal, the appropriate selection of a batch size remains elusive. The improved rate of convergence with increasing batch size suggests an optimal implementation should gradually increase the batch size from a minimum value to a maximum value. The extremes for these values would be two and *N*, respectively. Chapter 7 presents a method for determining an upper limit to the *Bsize* based on the relative entropy of the input data. A starting point for the minimum batch size might be estimated based on the relationship between the sampling frequency and an estimate of the largest frequency component of the input signal. Gradual increases in batch size might be triggered by improvements in relative entropy measures. However, since each epoch only produces an estimate of the actual performance, this may not be straightforward in practice. This problem remains a topic for future research.

The choice of an appropriate batch selection method is also dependent on the characteristics of the problem at hand in addition to the adaptation method applied to the learning system. In the noiseless time-series prediction problem with an appropriate sampling rate and GD adaptation, the sequential method proved superior. When advanced search techniques are used, random selection produced better results. When

noise is added, however, the terminal performance is improved by either random or representative batch selection. The oversampled case clearly showed the advantages of random and representative batch selection over the sequential method. Since batch training is based on the estimation of information performance metrics based on a subset of the original data, a representative selection of a data subset should provide a better quality estimate. Indeed the representative batch selection method has provided good results on an epoch-per-epoch basis. However, the computational overhead of representative selection reduces the overall efficiency of this approach when compared with random selection. Experimentally, random batch selection has proven to yield the most efficient and robust results for a wide range of problems when compared to representative and sequential selection.

Conclusions

The results presented in this chapter clearly demonstrate the advantages of applying batch training to ITL problems. Significant improvements in training efficiency were realized versus the traditional full ITL method. The SIG method was shown to be a special case of the batch training method. By exploring variations of the batch training parameter set, methods to improve the training efficiency of the SIG method were identified. The combination of batch training and advance parameter search methods provided further increases in efficiencies in the training process. Some guidelines for the proper selection of batch training parameters were presented. It should be noted, however, that the a priori selection of the optimum set of training parameters remains elusive. As with the training of many learning systems, the best results are often obtained by applying several solutions to the problem and selecting from among the best. The advances in computational efficiency presented in this and the previous chapters simply

allow these solutions to be generated much more efficiently. Because of these advances, the application of ITL techniques to problems with large data sets has been made to be a much more practical option than previously thought possible.

CHAPTER 7
PROPERTIES OF THE INFORMATION ESTIMATORS

Introduction

Chapters 4, 5, and 6 have proposed methods that expand the utility of ITL by addressing the computational complexity of both the criterion computation and the training process. In this chapter, the goal is to increase the utility of ITL by providing a better understanding of the information criteria and the parameters associated with its estimators. This chapter examines the estimators for entropy (RQE) and mutual information (CS-QMI and ED-QMI) in further detail and presents various properties and characteristics that help to define the effects and consequences of the various parameter settings. First, families of curves are used to provide useful insight as to the significance of each of the estimator parameters. These curves then help to derive and illustrate some useful properties of the estimators including a scale invariant version of the entropy estimator and a novel approach for the estimation of intrinsic dimensionality. An improved understanding of the characteristics of the estimators helps to place bounds on the estimator parameters and serves as a guide for selecting an appropriate parameter set based on the characteristics of the problem at hand.

Entropy Estimator

One of the difficulties in comparing or assessing the performance of a system trained with ITL stems from the performance criteria itself. When information-theoretic criteria are computed using Renyi's quadratic entropy and Parzen PDF estimation, the resulting values of entropy can have little absolute meaning. They only help gauge

performance in a relative sense when comparing data generated using the same set of parameters. Experience has shown, however, that the ability to change some of these parameters can greatly improve training results. For example, it has been shown that the kernel size, σ, plays an important role in the convergence of ITL systems and the ability to adapt the kernel size helps to avoid local extremes and assure better convergence [Erd02]. Adjustment of the sample size has also demonstrated improvements in training efficiency as demonstrated via the SIG and batch training methods. In other cases, the problem at hand might necessitate the comparison of datasets with different dimensionality or scale. For example, in the general problem of dimension reduction, the ability to compare the information content of feature sets with different dimensions (M) is critical.

Renyi's quadratic entropy estimator as described by (3-12) has served as the workhorse for much of the recent work in ITL research [Erd00, Erd01, Pri00, Xu98]. Despite its prevalence, little work has been published that examines the characteristics of the estimator and its dependencies on the parameters that characterize its performance. The following sections help to explore the relationships between the entropy estimator and its major parameters by using a family of curves approach combined with some pertinent examples.

Families of Entropy Curves

The primary variable parameters of the entropy estimator that can be user defined or are a function of the problem at hand are the following:

1. N - The number of data samples or exemplars.
2. σ - The kernel size for the Parzen PDF estimate.
3. M - The dimension of the data set.

4. d – A measure of the extent (or variance) of the data.

Figures 7-1, 7-2, and 7-3 show the dependence of the entropy estimate on these parameters and highlight some of the difficulties with the entropy criterion estimator. These figures were generated by estimating Renyi's quadratic entropy with Parzen PDF estimation from a random M-dimensional, uniformly distributed (over the range $[0,d]$) dataset of N samples.

Notice that the entropy value can take on virtually any value depending on the settings of the parameters. Each shows the entropy estimate, H_{R2E} from (3-12) in red, versus the square of the kernel size, σ^2, of the random uniformly distributed dataset for a variety of N, M, and d combinations. Note the extreme variability of entropy due to the different parameters for the exact same underlying distribution.

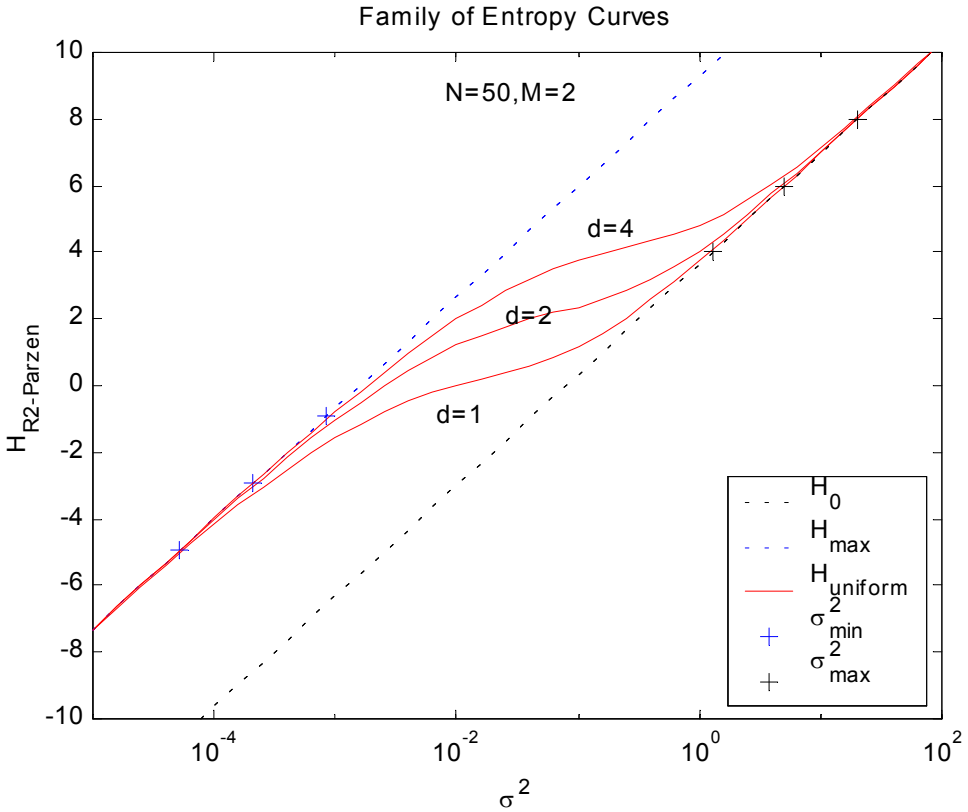

Figure 7-1. Family of entropy curves: Variable scale.

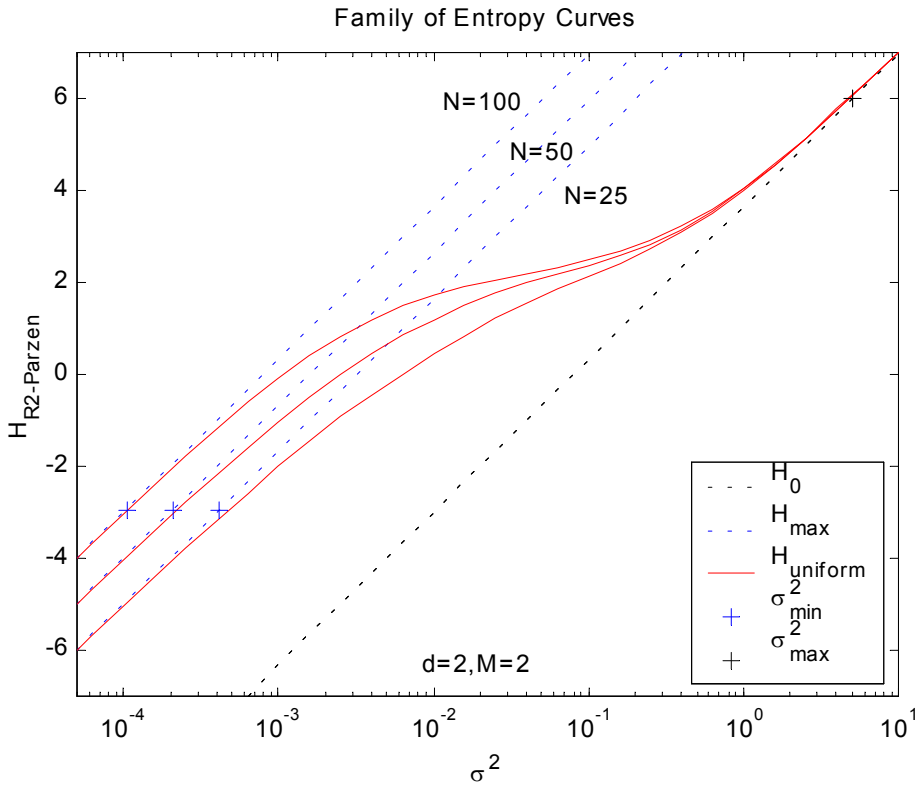

Figure 7-2. Family of entropy curves: Variable sample size.

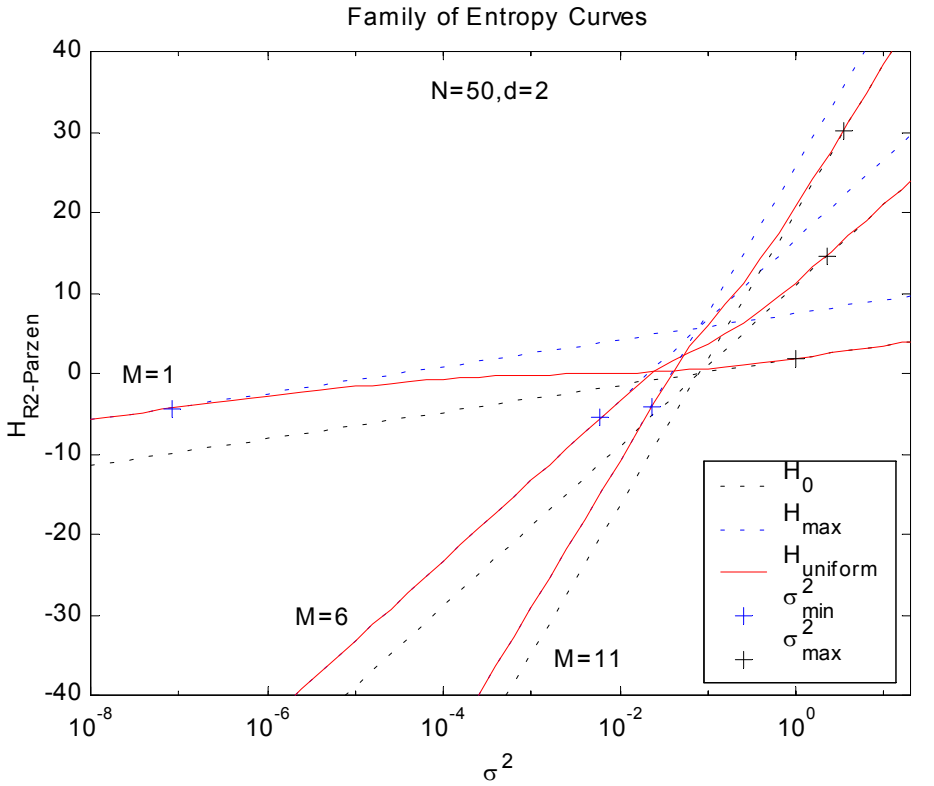

Figure 7-3. Family of entropy curves: Variable dimensionality.

Also of interest is that in all cases, the entropy estimate asymptotically approaches one line (in black) as $\sigma \to \infty$ and another parallel line (in blue) as $\sigma \to 0$ for a given set of parameters, N, M, and d. The first line, denoted $H_0(\sigma)$, is a lower bound on H_{R2E} as a function of σ and M. The second is an upper bound on H_{R2E} as a function of σ, M, and N. The characteristics of these lines can be solved analytically as,

$$H_0(\sigma) = \lim_{\sigma \to \infty} H_{R2} = \frac{M}{2}\log 4\pi + \frac{M}{2}\log \sigma^2 \tag{7-1}$$

$$H_{\max}(\sigma) = \lim_{\sigma \to 0} H_{R2} = \frac{M}{2}\log 4\pi + \frac{M}{2}\log \sigma^2 + \log N \tag{7-2}$$

Equation (7-2) assumes that the N data points are unique. In some applications, such as with discrete or saturated data, some of the data points might be repeated. If this condition exists, the maximum equation in (7-2) can be modified to provide a more accurate maximum as follows:

$$H_{\max}(\sigma) = \lim_{\sigma \to 0} H_{R2} = \frac{M}{2}\log 4\pi + \frac{M}{2}\log \sigma^2 + \log \frac{N^2}{N + \sum_{non-unique} n_i!} \tag{7-3}$$

where n_i represents the number of times each nonunique data sample exists in the dataset. There are some cases where nonunique data is acceptable or expected and the adjusted maximum from (7-3) should be used to determine the maximum entropy for the dataset at hand. However, in other cases, the repeated data might be undesirable in general perhaps resulting from the underlying statistics of the dataset and the current state of the mapping function and the maximum from (7-2) provides a better basis for comparison with other datasets.

In (7-1), as $\sigma \to \infty$, the kernel size becomes so large that the information forces between data samples are indistinguishable. From the perspective of the kernel forces, all

points are effectively collocated and contribute equally to the information potential. The net effect is that the entropy estimate saturates and the information forces approach zero. Conversely, in (7-2) as $\sigma \rightarrow 0$, a point is reached where there is virtually no interaction between data samples because of the small kernel size. This corresponds to the minimum IP where the only contribution is from the number of points, N, in the space. In this case, the information forces approach zero as well. The red lines can be interpreted as an approximation of the maximum entropy level that can be achieved at a particular σ for the corresponding M, N, and d.

It is important to note the distinction between the estimation, $H_{R2E}(\sigma,N)$, and the actual entropy value, $H_{R2}(x)$. The former is a functional estimate of the later that is dependent on the number of samples, N, taken from the random variable, $\boldsymbol{X}$, and the kernel size, σ, used for PDF estimation. The quantities $H_0(\sigma)$ and $H_{\max}(\sigma,N)$ described in (7-1), (7-2), and (7-3) are also byproducts of the estimation of the actual entropy using Parzen PDF estimation.

By examining the figures more closely, it becomes evident that the properties of Renyi's quadratic entropy with Parzen PDF estimation can pose problems for adaptation of the kernel size, particularly for large M. Note that if the kernel size starts large and is gradually decreased, the entropy value will tend to decrease because of the slope of the asymptotic boundaries on H. This could cause problems for many parameter search routines seeking to maximize entropy where the kernel size is allowed to adapt or if the kernel size itself one of the optimization parameters. The adaptive kernel size modifications for parameter search presented in Chapter 5 address these issues and help to compensate for this phenomenon.

Kernel Size Ranges

The first and perhaps most obvious application stemming from the examination of these curves is to help determine appropriate values of the kernel size in a given problem. The curves clearly show the saturation that occurs for both large and small kernel sizes as the entropy estimate approaches the asymptotic limits defined by H_0 and $H_{\max}$, respectively. By constraining the kernel size to be within an appropriate region, the training process can avoid the slow learning or stagnation that can occur when the entropy estimation becomes saturated. Upper and lower limits on the kernel size can be used as limits during adaptation for an adaptive kernel size implementation. They can also serve as the start and end points for the kernel annealing process described by Erdogmus, et al.

In order to develop an analytical expression for the kernel size limits, a combination of an intuitive approach and empirical trial and error is presented. First, intuition suggests that the kernel limits be proportional to the extent, d, of the data. In determining a maximum kernel size for M=1, a conservative approach is taken by equating the kernel size to the extent of the data (i.e., $\sigma_{\max}=d$). This approach ensures kernel interaction forces to be present throughout the extent of the sample space. In order to extend this to any dimension, M, it is desirable to take into account the effect of the decreasing relative volume of a hypersphere inset within a hypercube so that interaction forces can be present even in the corners of the hypercube. The ratio of the m-dimensional volume of a hypersphere and a hypercube is given by (7-4).

$$\frac{vol(Sphere_m)}{vol(Cube_m)} = \left(\frac{\sqrt{\pi}}{2}\right)\frac{1}{\Gamma(\frac{m}{2}+1)} \tag{7-4}$$

By using (7-4) to scale the kernel size, the expression in (7-5) for maximum kernel size is obtained.

$$\sigma_{\max}^2 = \frac{4d^2}{\pi}\Gamma(\tfrac{M}{2}+1)^{2/M} \tag{7-5}$$

Next, for a minimum kernel size, the idea is to select an expression that determines a kernel size below which there is likely to be little or no interactions and consequently little benefit to attempt training. Since the curves above were generated for uniformly distributed data, tighter clustering of data might require smaller kernel sizes. For these cases, an expression is sought that provides a viable minimum even when the data is highly clustered in the sample space. Although the expression in (7-6) was determined empirically by trial and error in order to arrive at an analytical expression for the kernel size at the saturation point for $H_{\max}$, it can be explained intuitively as follows. First, the nearest spacing for N evenly distributed points in an M dimensional hypercube is determined. The kernel size is then set to a fraction of this distance, in this case 1/7, for very small kernel interaction. This distance is scaled so that the data is concentrated to cover only 10% of the total sample space for relatively tight clustering. Finally, the distance is scaled according to (7-4) resulting in the expression described by (7-6) for a minimum kernel size.

$$\sigma_{\min}^2 = \left(\frac{d}{7N^{1/M}}\right)^2 \frac{4}{\pi}\left[0.1\times\Gamma(\tfrac{M}{2}+1)\right]^{2/M} \tag{7-6}$$

The black and blue crosses in Figures 7-1, 7-2, and 7-3 represent the evaluation of the kernel size maximums and minimums from (7-5) and (7-6), respectively. It is evident from the figures that, at least for the case of uniform data, these expressions closely track the saturation points for both H_0 and $H_{\max}$.

As noted previously, the expressions for kernel size range in (7-5) and (7-6) have been developed from families of curves derived from uniformly distributed data. As such, they are probably well suited for entropy maximization problems. For entropy minimization, the minimum kernel size might be required to be made much smaller for better optimization. In addition, if during initial training the data set is highly clustered, a smaller kernel size may be required in order to begin the separation process for maximum entropy problems. In other words, the expressions presented here simply establish some basic guidelines for kernel size selection; care must be taken by exploring the characteristics of the data prior to applying any expressions arbitrarily.

Effects of Different Scale and a Scale-Invariant Entropy

Scale invariance is a desirable characteristic for many cost functions since it allows valid comparisons between data sets without the need for normalization. The ITL entropy criteria, however, does not possess the characteristic of scale invariance. For some problems, preprocessing of the data or the topology of the learning system requires different scaling of the data. In order to fairly compare the entropy of data sets with different scaling a closer inspection of Figure 7-1 is required. This figure shows the change in entropy due to a change in scale of the dataset. For d=1, the uniformly distributed random numbers are generated on the interval from 0 to 1 in all M dimensions. For d=2 and d=4, the original data is simply multiplied by the corresponding factor, d. This scaling produces an entropy curve of the same shape as the original that is translated by a factor of d_2/d_1 in σ and by the corresponding slope of H_0 or H_{max} in entropy. Therefore, in order to translate an entropy estimate in one data scale and kernel size (d_1, σ_1), to another scale (d_2, σ_2) for comparison, the translation formula in (7-7) can be applied. The proof follows in the paragraphs below.

$$H(d_2,\sigma_2) = H(d_1,\sigma_1) - H_0(\sigma_1) + H_0(\sigma_2)$$
$$H(d_2,\sigma_2) = H(d_1,\sigma_1) + M\log\frac{d_2}{d_1} \quad \text{, where } \sigma_2 = \frac{d_2}{d_1}\sigma_1 \tag{7-7}$$

Note that the kernel size must also be scaled in proportion to the scale of the data in order to maintain the same relative position on the entropy curve. This makes sense since the amount of interaction in the entropy computation is proportional to the ratio between the standard deviation or scale of the data and the kernel size. Figure 7-4 illustrates the scaling approach and verifies that this method can be applied to compare and translate entropies with different scaling factors. In this figure, the original entropy curves from Figure 7-1 are translated by various scale factors. The curves denoted 'd_A to d_B' use the translation formula in (7-7) to shift data scaled on the range [0,A] to a range of [0,B]. As is evident, the translated entropy curves line up accurately with the curves generated with the original data with the corresponding data scale.

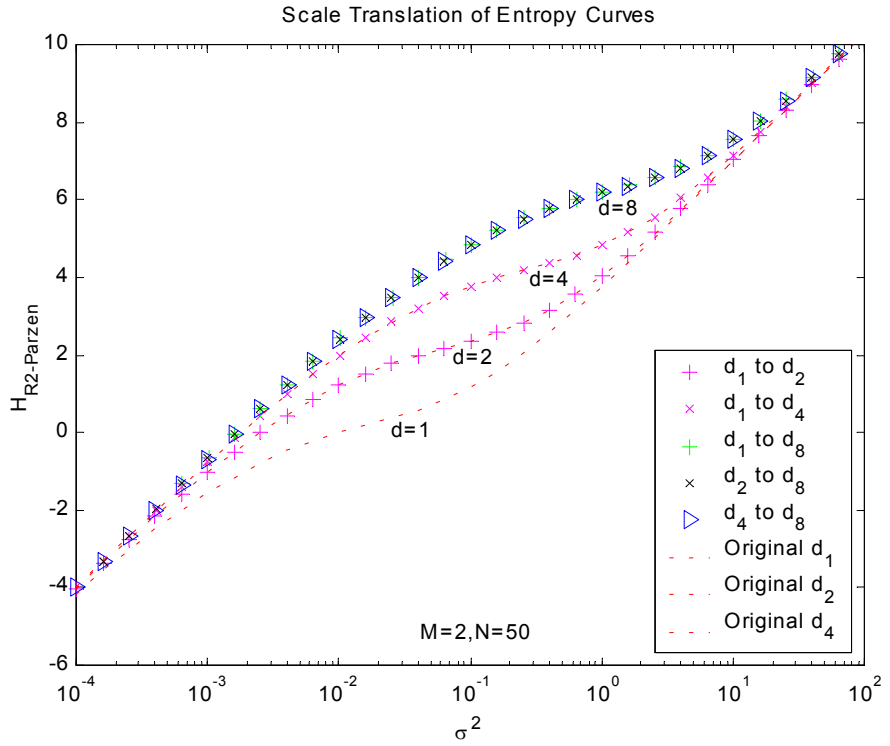

Figure 7-4. Scale translation examples.

Proof of Scaling Equation. Recalling that the Parzen PDF estimation with Gaussian symmetric kernels defined by (3-10) and (3-11) when combined with Renyi's entropy definition from (3-4), results in the following estimate for Renyi's quadratic entropy,

$$H_{R2}(y,\sigma) = -\log\left[\frac{1}{N^2}\sum_i\sum_j G(\Delta y_{ij}, 2\sigma^2 I)\right]$$
$$H_{R2}(y,\sigma) = -\log\left[\frac{1}{N^2}\sum_i\sum_j \frac{1}{\left(4\pi\sigma^2\right)^{M/2}} e^{\frac{-(\Delta y_{ij})^T(\Delta y_{ij})}{4\sigma^2}}\right] \tag{7-8}$$

If the original data, y, and kernel size, σ, are scaled by a factor of a, we obtain,

$$H_{R2}(ay, a\sigma) = -\log\left[\frac{1}{N^2}\sum_i\sum_j G(a\Delta y_{ij}, a^2 2\sigma^2 I)\right]$$
$$H_{R2}(ay, a\sigma) = -\log\left[\frac{1}{N^2}\sum_i\sum_j \frac{1}{\left(4\pi a^2\sigma^2\right)^{M/2}} e^{\frac{-(a\Delta y_{ij})^T(a\Delta y_{ij})}{4a^2\sigma^2}}\right] \tag{7-9}$$

Noting that the a terms in the exponential cancel and factoring the a terms out of the sum yields,

$$H_{R2}(ay, a\sigma) = -\log\left[\frac{1}{|a|^M}\frac{1}{N^2}\sum_i\sum_j \frac{1}{\left(4\pi\sigma^2\right)^{M/2}} e^{\frac{-(\Delta y_{ij})^T(\Delta y_{ij})}{4\sigma^2}}\right] \tag{7-10}$$

Utilizing the properties of logarithms of products yields

$$H_{R2}(ay, a\sigma) = -\log\left[\frac{1}{N^2}\sum_i\sum_j \frac{1}{\left(4\pi\sigma^2\right)^{M/2}} e^{\frac{-(\Delta y_{ij})^T(\Delta y_{ij})}{4\sigma^2}}\right] - \log\left(\frac{1}{|a|^M}\right) \tag{7-11}$$

and finally, the desired result.

$$H_{R2}(ay, a\sigma) = H_{R2}(y,\sigma) + M\log|a| \tag{7-12}$$

The result in (7-12) suggests that a scale invariant criterion based on Renyi's quadratic entropy might be,

$$J_{R2}(y,\sigma) = H_{R2}(y,\sigma) - \frac{M}{2}\log[\mathrm{var}(y)] \tag{7-13}$$

with the condition that the kernel size be directly proportional to the spread of the data as,

$$\sigma^2(y) = k\,\mathrm{var}(y) \tag{7-14}$$

where k is a constant.

Sample Size Variations, Relative Entropy and Batch Size Estimation

Figure 7-2 shows the increase in the entropy estimate with increased sample size for a given σ, implying that use of the entire data set is necessary to make use of all information. However, batch training successes have demonstrated that there is a point of diminishing returns at least in terms of training efficiency. Intuitively, the entropy of a specific distribution should not depend on the number of samples taken from it. For example, 64 samples of a uniform distribution should have a comparable entropy value to that of 256 samples of a uniform distribution. Figure 7-5, shows that this is the case for a family of horizontal lines requiring each to have a unique σ as a function of N. Of particular interest are the kernel sizes defined by the magenta triangles. These triangles correspond to the theoretical maximum entropy of a range-limited signal (i.e., the entropy of a uniform distribution) using Renyi's entropy measure directly from (3-4) with $\alpha = 2$ as described by (7-15). It should be noted that analytical solutions for (3-4) only exist in special cases; however, since the uniform distribution provides the maximum entropy for a range-limited signal, it is a useful special case.

$$H_{R2\max} = -\log\left(\int_{-\infty}^{\infty} u(\mathbf{z})^2\,d\mathbf{z}\right) = M\log d \tag{7-15}$$

The resulting kernel sizes correspond to the solution of equating the theoretical maximum with the estimator maximum, or,

$$H_{R2\max} = H_{\max} \tag{7-16}$$

that yields,

$$\sigma_i^{\,2} = \frac{d^2}{4\pi N_i^{\,2/M}} \tag{7-17}$$

Equation (7-17) suggests a kernel size as a function of N that might be used to compare entropy estimates computed with different sample sizes.

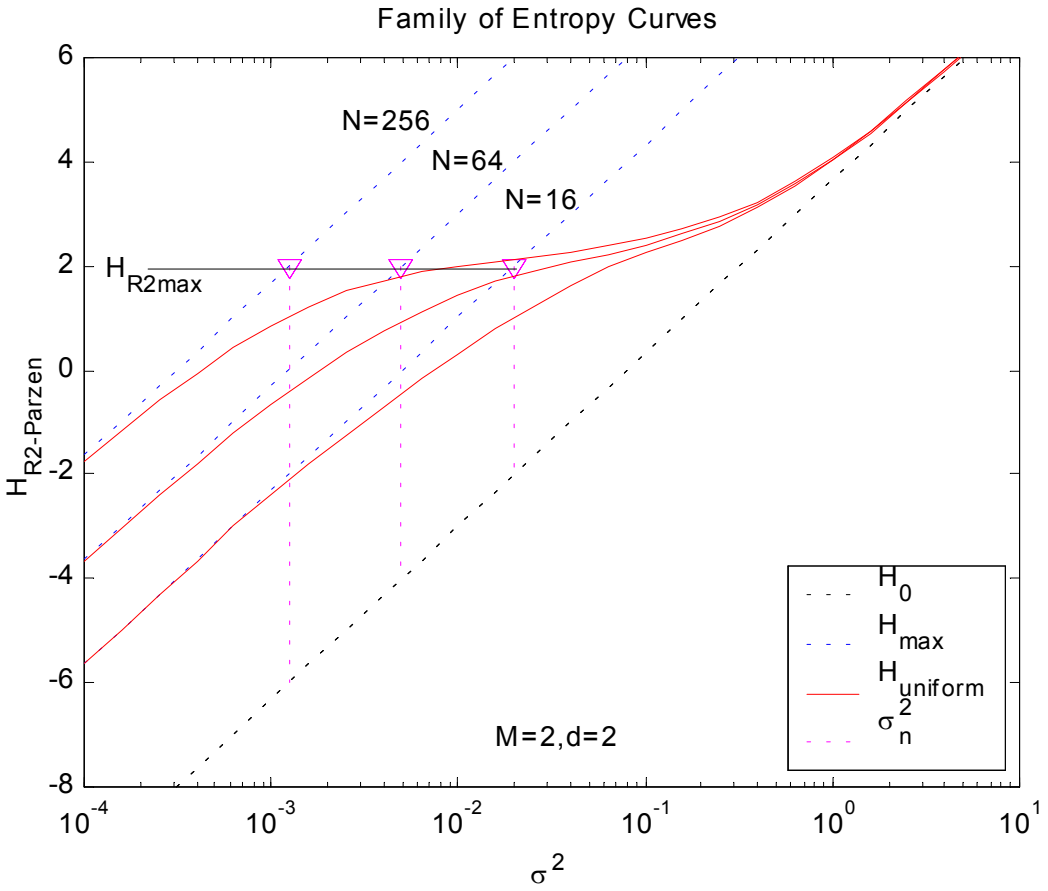

Figure 7-5. Family of entropy curves: Kernel size equivalents.

Note that (7-16) equates the maximum value of the Renyi's quadratic entropy from (3-4) and (7-15) with the maximum of the estimator in (3-12) and (7-2) resulting in a specific kernel size for a given N, M, and d. Using these kernel sizes, a novel metric for comparison of entropy estimates with different samples sizes and the same

dimensionality (i.e., $H_{N_1}(N_1,\sigma_1)$ versus $H_{N_2}(N_2,\sigma_2)$) is proposed. Consider the relative entropy measure described by (7-18).

$$H_{relative}(N_n) = \frac{H_{N_n}(N_n,\sigma_n) - H_0(\sigma_n)}{H_{\max}(\sigma_n) - H_0(\sigma_n)} \tag{7-18}$$

In (7-18) the estimated entropy, H_N, is computed using the kernel size from (7-17) for the corresponding value of N. It is then translated by the minimum of the estimator, H_0, and scaled by the maximum range of the estimator $H_{R2\max}$ -H_0. The net result is a relative entropy estimate that can take on values in the range [0,1]. This estimate provides a basis for comparison of entropies computed using a different number of sample points. Furthermore, if a kernel size other that (7-17) is used, perhaps because of the spread of the data, as long as the ratio from σ_1 to σ_2 follows the relationship for N in (7-17), valid comparisons can be made.

Figure 7-6 shows an example of the relative entropy metric for a uniformly distributed random dataset that has been sampled with various values of N. The curves show the consistency of the relative entropy, aside from the expected fluctuations for very small N, across a wide range of sample sizes.

Another useful application of the relative entropy metric is motivated by the batch training method described in Chapter 6. For the batch method, the determination of an appropriate batch size is a key design parameter for batch implementations. In order to facilitate the batch size selection, the relative entropy metric is used to examine the information content as a function of increasing sample size. The examples that follow show the approach for several batch-training problems.

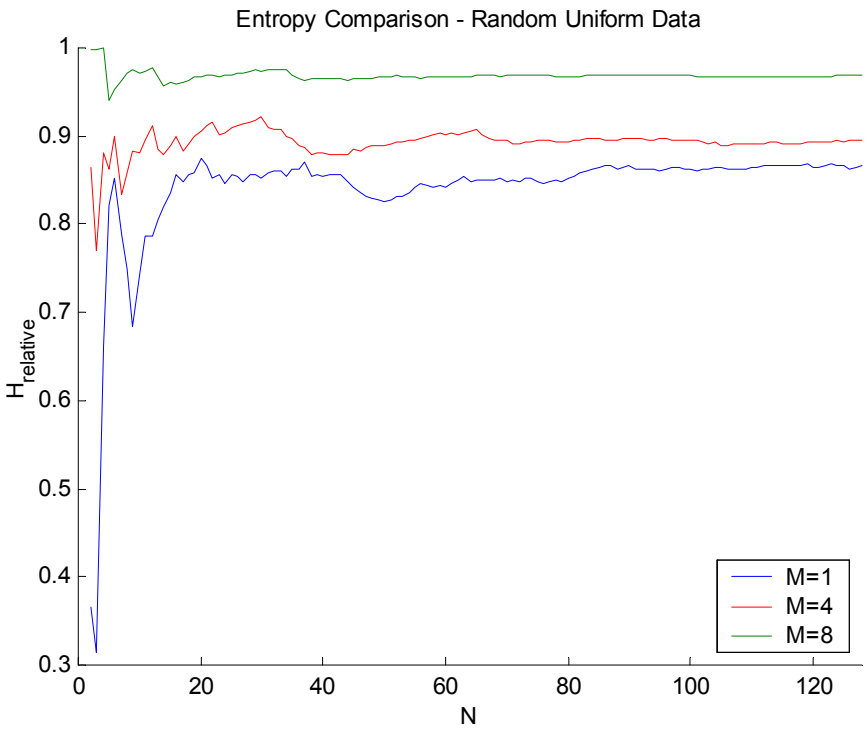

Figure 7-6. Relative entropy comparison across variable sample sizes.

Figure 7-7 shows the relative entropy comparison for two cases. The first example is N=100 samples of a one-dimensional mixture of two sinusoids with a composite period of approximately 50 samples. In this case, the relative entropy reaches a maximum at N=50, as might be expected, and begins to level off around N=30. This suggests that batch training could be conducted with N<<100 with little loss of terminal accuracy. The second example is the time-series prediction problem presented in Chapter 6. Recall that in this example, the input data set consists of N=32 samples of a composite sinusoidal signal. Each of these samples is a two dimensional vector consisting of sequential signal samples to be used to predict the next sample. In this case, the relative entropy reaches a plateau in the N=8-14 sample range, suggesting a batch implementation of this size. Recall that previous experimentation with various batch sizes on this problem yielded excellent results for batch sizes of 8 and 16.

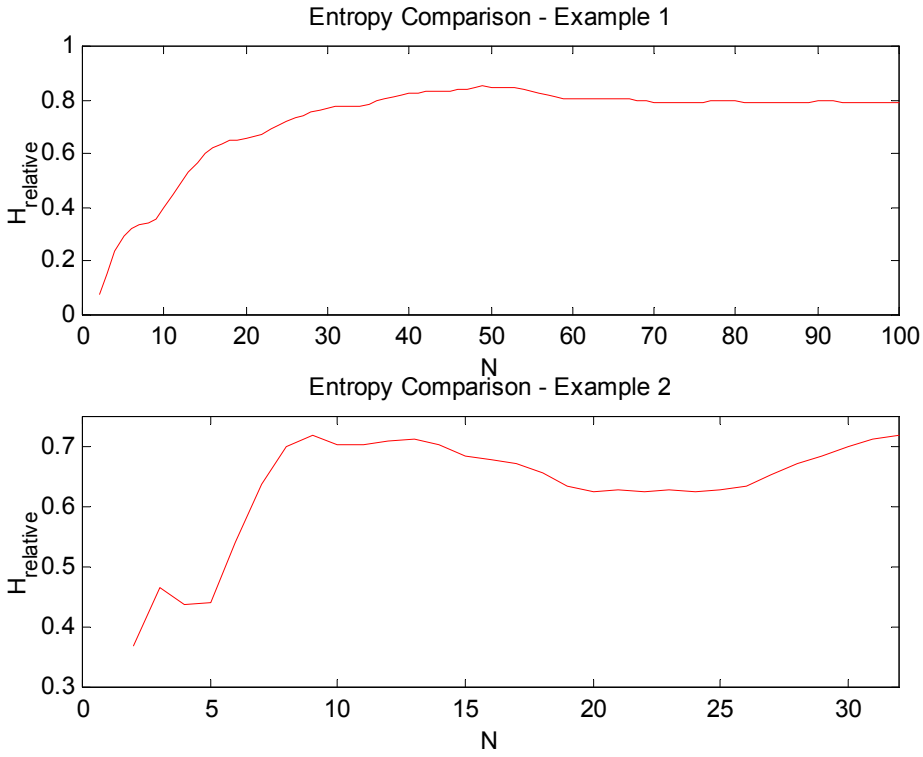

Figure 7-7. Relative entropy comparison for sample problems.

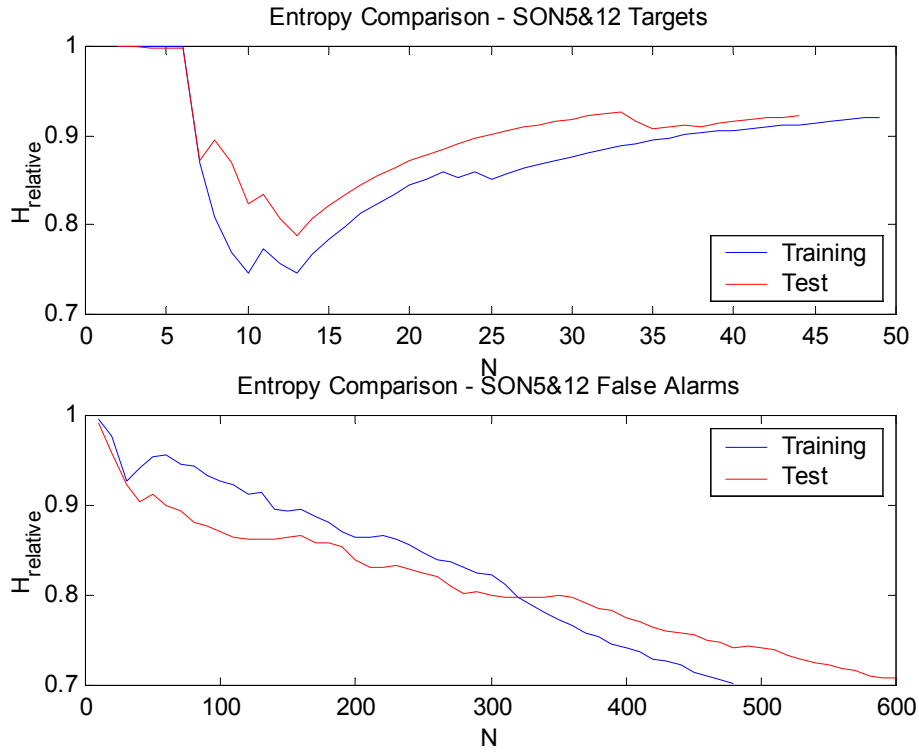

Figure 7-8. Relative entropy comparison for SON512.

Figure 7-8 shows a third example using the Sonar 512 feature data set. This figure shows the relative entropy comparison for the target class and false alarm class for both the training set (in blue) and the test set (in red). Note here that for both the training and test set, the relative entropy of the target class continues to increase with the addition of new targets implying that most targets are providing relevant new information. In contrast, the relative entropy for the false alarm class begins to decrease steadily after the first 50 samples suggesting that not all of the false alarms might be necessary during training.

Entropy-Based Intrinsic Dimensionality Estimators

Figure 7-3 shows the effects of increasing dimensionality on the entropy estimator and hints at the difficulties in comparing entropies of different dimension. In this section, these effects are examined so that they can be exploited in order to extract the underlying dimensionality information from a given dataset. Estimation of dimensionality has a wide range of application including feature extraction and data compression. An accurate estimate of dimensionality helps in determining the number of variables needed to represent a dataset. This knowledge can help establish the topology of a mapping function, and in the context of ITL, helps to define or constrain the design parameter M. The paragraphs below present two approaches for intrinsic dimensionality estimation based on entropy. The first approach represents a novel method developed through the conduct of this research based on the relative entropy definition of (7-18). The second approach is closely related to method proposed by Grassberger and Procaccia for characterization of strange attractors [Gra83].

Relative entropy intrinsic dimensionality estimator

In the context of information theory, entropy has been described as a measure of information content as in Chapter 3. The same definition of entropy can also be interpreted as a measure of the average number of bits or symbols required for representing a random variable. Random variables with larger entropy require more bits, or variables, to represent them. Similarly, datasets with higher intrinsic dimensionality require more symbols or variables to represent them. Based on this interpretation of entropy and its parallelism with dimensionality, it is conjectured that entropy can be used as a proxy for determination of dimensionality. A novel approach that depends on the relative entropy metric described previously is proposed.

Consider a dataset of N samples with an apparent dimensionality of M_{max}. It is desired to find the intrinsic dimensionality, M_{ID} of the original dataset. The relative entropy approach is described as follows: First, a calibration dataset comprised of a uniform distribution of M_{max} dimension and N samples is used to generate a relative entropy curve as a function of intrinsic dimensionality, M_{ID}, for all values of M_{ID} from 1 to M_{max} as follows. For $M_{ID}=M_{max}$, each dimension of the calibration dataset distribution is an independent uniform random variable. As M_i is decremented, the overall dimension of the calibration dataset remains M_{max}, however, an additional dimension (or feature) of the dataset is forced to be dependent on the other features, thus reducing the intrinsic dimensionality, but maintaining the total dimension of the calibration dataset so that comparisons can be made. For $M_{ID}=1$, all M_{max} features contain the same data. The result is a relative entropy curve that models the relative entropy of a dataset as a function of M_{ID} intrinsic dimensionality embedded in M_{max} dimensional data. Finally, the relative entropy of the original dataset is computed and compared with the relative entropy curve.

The point on the abscissa, or M_{ID} axis, at which the corresponding relative entropy curve equals the relative entropy of the original data set yields an estimate of the intrinsic dimensionality of the original dataset. The detailed example below shows the process.

The M_{max} and N values for the SON512 training set are 60 and 538, respectively. A relative entropy curve is generated using uniform random data with from 1 to 60 independent features embedded in 60 dimensions. This is shown in red in Figure 7-9. The relative entropy of the SON512 training set is then computed to be 0.71 and is shown as the blue line in the figure. The point of intersection, which occurs at approximately M_{ID}=6.7 in this case, can be regarded as an estimate of the intrinsic dimensionality of the dataset. For corroboration, Figure 7-10 shows the PCA residual errors versus number of features for the same dataset. Typically, PCA overestimates dimensionality because of the limitations of being a linear technique. The residual errors for M=7 are 2%. Although this does not necessarily validate the technique, the results are at least consistent with the relative entropy approach presented here. Additional examples are presented in the sections that follow.

Information potential dimensionality estimator

Grassberger and Procaccia introduced a measure for characterizing strange attractors based on the correlations between random samples of points in the space of the attractor. Using a correlation integral as defined by,

$$C(l) = \lim_{N\to\infty} \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \theta\,(l - |\,\mathbf{X}_i - \mathbf{X}_j\,|) \tag{7-19}$$

where $\theta\,(x)$ is the Heaviside function, the behavior of (7-19) is shown to be a power of l for small l as in (7-20).

$$C(l) \propto l^{v} \tag{7-20}$$

The exponent, v, in (7-20) is referred to as the correlation dimension and provides a lower bound on fractal dimension of the attractor.

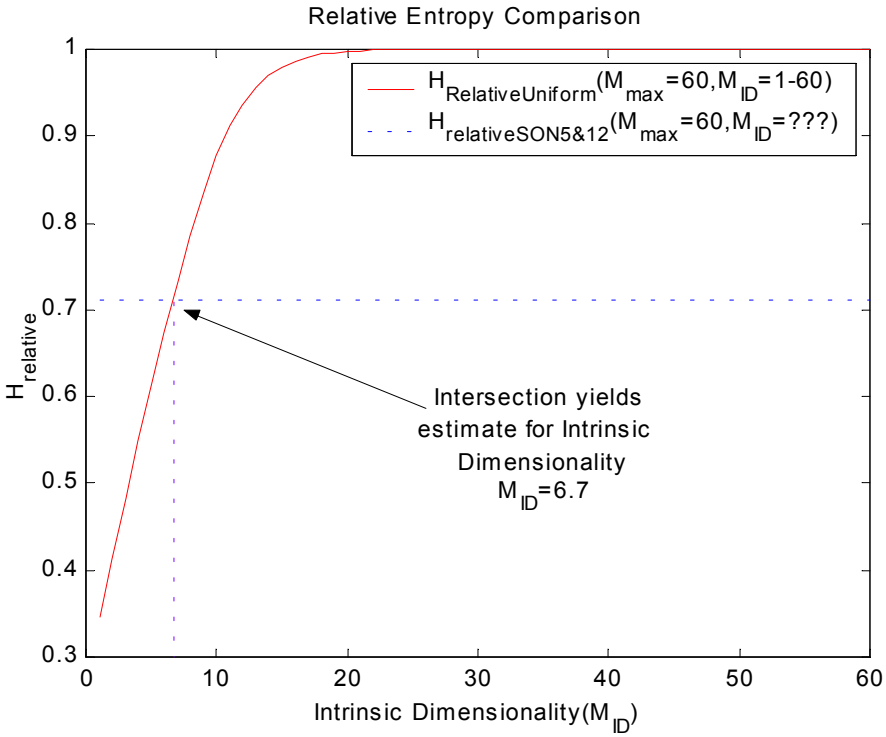

Figure 7-9. Relative entropy ID estimation for SON512 training set.

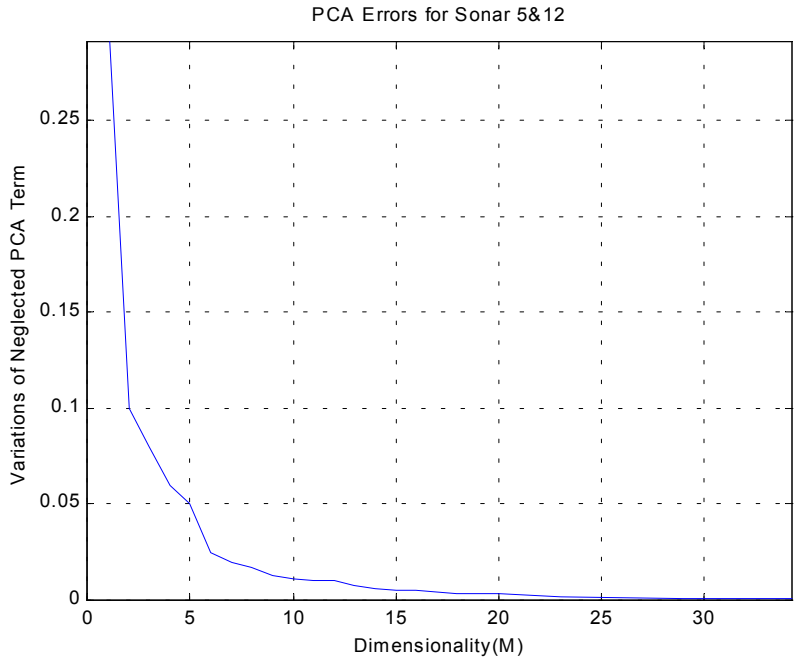

Figure 7-10. The PCA residual errors for SON512 training set.

Note the similarity between the correlation integral of (7-19) and the information potential of (3-12). The information potential is essentially a smoothed version of the correlation integral that uses a Gaussian kernel as compared with the discrete Heaviside function. Here the distance threshold parameter, l, is analogous to the kernel size, σ, both of which control the number or amount of interactions for the summation. Noting these similarities, the curves in Figure 7-3 are plotted again in Figure 7-11. However, in this case, they have been normalized by subtracting the corresponding H_0 to eliminate the slope cause by the σ and M terms in the denominator of the Gaussian kernel. Of interest here is the increasing slope of the transition of the curves with increasing M. This corresponds with the observations of Grassberger and Procaccia and consequently, can provide an alternative measure of correlation dimension.

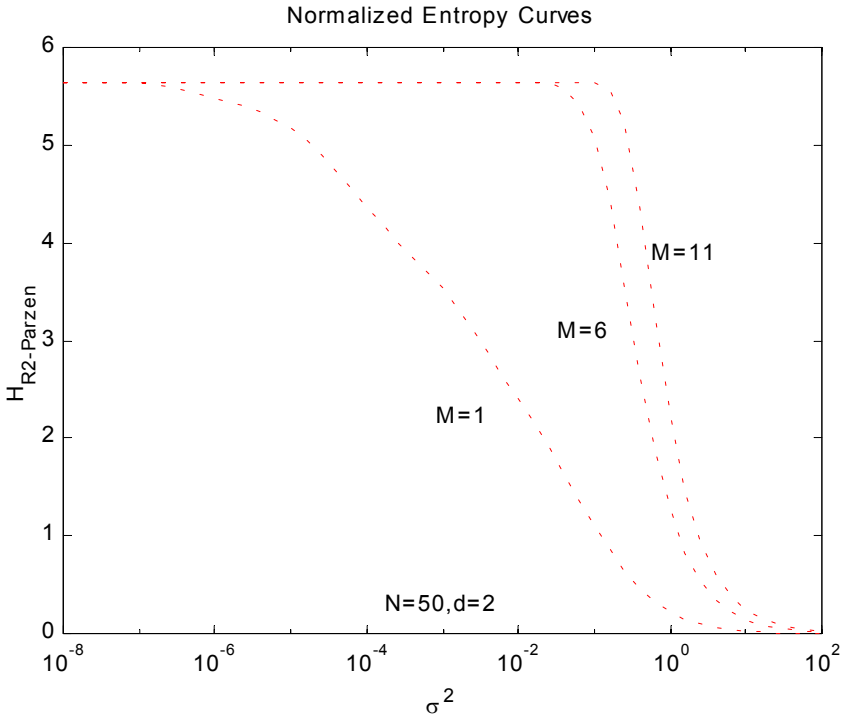

Figure 7-11. Dimension-normalized entropy curves.

Figure 7-12 shows the process for estimating dimensionality with both the Grassberger-Procaccia method and the normalized information potential. The blue

curves plot the logarithm of the correlation integral and the estimate of the correlation dimension versus the radius, l. The correlation dimension is determined by estimating the slope of the straight portion of the correlation integral transient. The red curves show the logarithm of the normalized information potential and associated dimensionality estimate versus the kernel size, σ. The data for this example consists of 1000 points of random uniformly distributed data in three dimensions so the expected dimensionality is three. Notice that the information potential provides a smoother slope estimate due to the Gaussian kernels. However, the smoothing process also reduces the length of the straight portion of the curve along with the magnitude of the slope estimate. For this example the Grassberger-Procaccia method estimates M_{ID}=2.70 while the information potential method estimates M_{ID}=2.57.

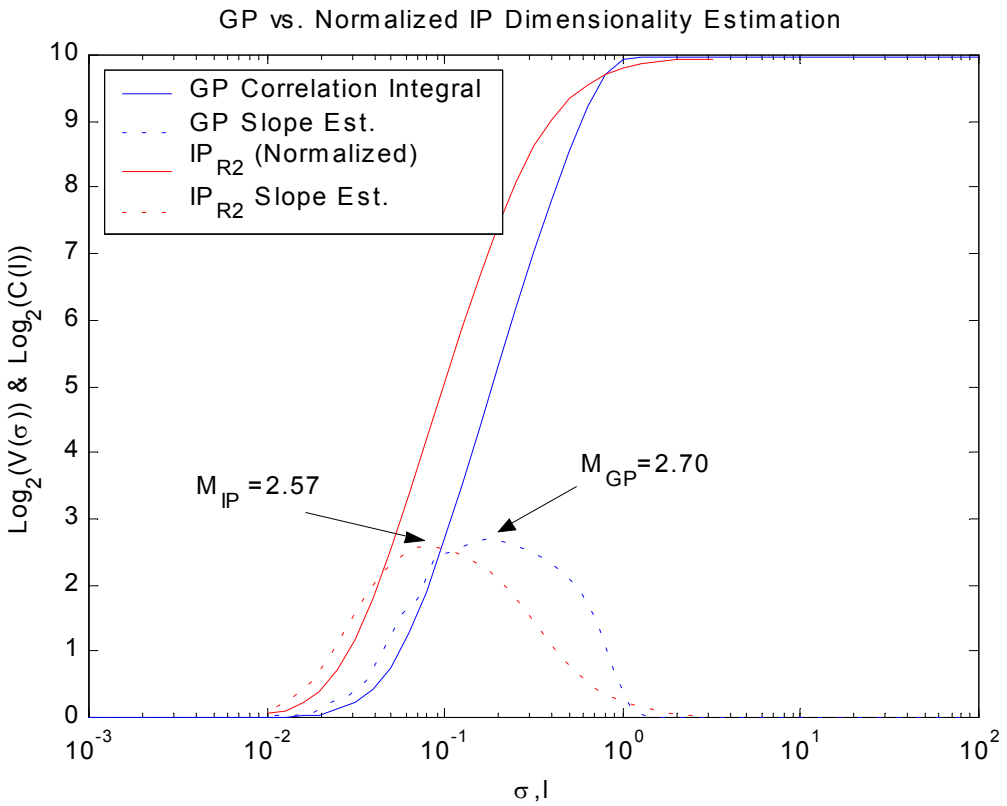

Figure 7-12. Grassberger-Procaccia and IP dimension estimators.

Dimensionality estimation examples

In this section, the dimensionality estimators described previously are applied to a variety of sample problems for comparison. In addition the to the Relative Entropy (RE), Grassberger-Procaccia (GP), and Information Potential (IP) estimators, the K-Nearest Neighbor (KNN) approach for dimensionality estimation is also applied [Fuk90]. Table 7-1 compares the estimate values with the known values for several known problems in addition to the SON512 dataset.

Table 7-1. Comparison of dimensionality estimators.

Data Description			Dimensionality Estimates			
Problem Name	N	M_{ID}	M_{KNN}	M_{GP}	M_{IP}	M_{RE}
Henon Map (a=1.4 b=0.3)	2000	1.26	1.23	1.15	1.15	1.30
Kaplan-Yorke Map (a=0.2)	1000	1.431	1.41	1.30	1.27	1.36
Logistic Equation (b=3.5699456…)	1000	0.538	-	0.52	0.49	0.56
Lorentz Equation	1000	2.06	1.89	1.77	1.71	1.96
1D Uniform	1000	1	0.94	0.96	0.96	0.95
2D Uniform	1000	2	1.80	1.84	1.81	1.99
4D Uniform	1000	4	3.29	3.31	3.11	3.99
8D Uniform	1000	8	5.97	5.93	4.97	7.97
16D Uniform	1000	16	10.45	9.84	6.48	15.95
Sonar 512 Train Set	538	?	6.04	5.34	3.79	6.75
Sonar 512 Test Set	644	?	5.58	5.46	3.99	6.89
Sonar 10 Train Set	481	?	6.05	4.20	3.31	6.89
Sonar 10 Test Set	566	?	6.17	4.42	3.27	6.36

Notice that the IP estimate is consistently less than the GP estimate because of the smoothing of the Gaussian kernels. Since the GP method provides a lower bound on the dimensionality, and the computational complexity of the IP method is significantly

greater than that for GP, the practical utility of the IP method is not clear. The similarity between the IP and the GP method, however, reinforces the inferred relationship between entropy and dimensionality.

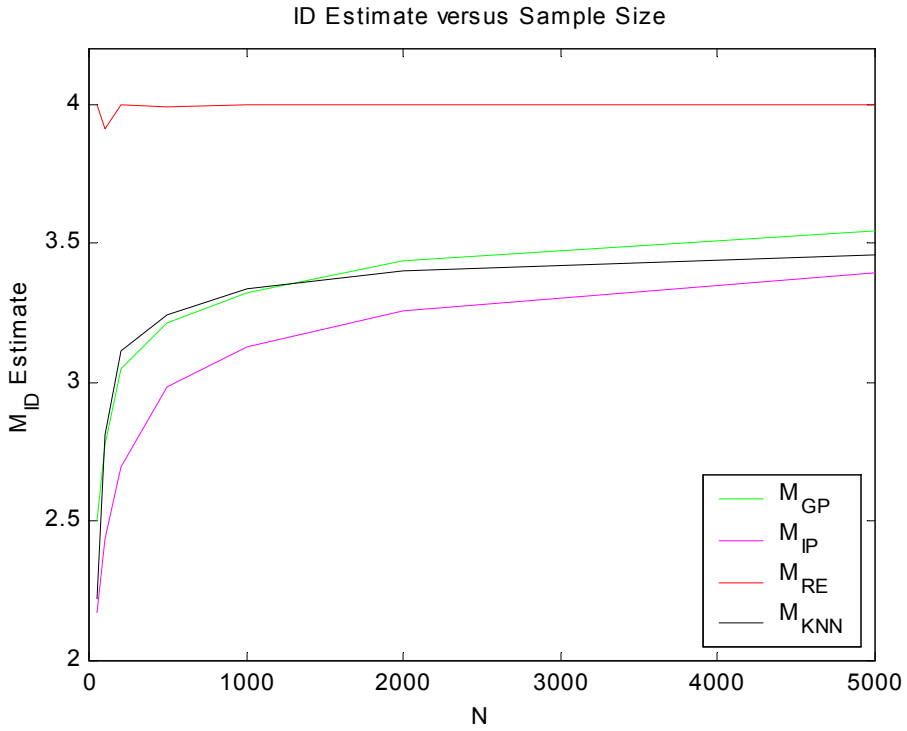

Figure 7-13. Dimension estimation dependency on sample size.

The RE estimator, on the other hand, can provide useful information with respect to dataset dimensionality. The strength of the method is apparent in the ID estimation of higher dimensional datasets. Because the RE method uses a calibration dataset of the same size, *N*, as the original dataset, it provides much closer estimates for sparsely populated high dimensional spaces. This allows for a better estimate of an upper bound on the number of variables need to represent a dataset than using either the GP or KNN methods when the number of samples is limited. Figure 7-13 shows the ID estimator performances with increasing sample size for a four-dimensional uniform dataset. Note that the GP, IP, and KNN methods significantly underestimate the dimensionality for this

range of sample sizes. Camastra and Vinciarelli proposed a variation on the GP method that also uses a calibration dataset for ID estimation [Cam00]. Also of interest is that for the fractal datasets in Table 7-1, the RE can either overestimate or underestimate the true dimensionality. From this perspective, it is concluded that the RE method cannot be used as either an upper or lower bound for ID estimation.

Mutual Information Estimators

Like the entropy criterion, the CS-QMI and ED-QMI estimators for mutual information of (3-16) and (3-17) are burdened with similar difficulties when comparing or assessing the performance of systems trained with these criteria. Since these estimates lack a consistent absolute interpretation, they can only measure performance in a relative sense when compared to data using the same set of parameters. To complicate matters in this case, the functional form of these estimators is more complex than the entropy estimator since mutual information measures the similarity between two or more signals rather than estimating a property of a single signal. Although this complexity makes generalizations more difficult, the following sections attempt to provide some insight to the relationships between these mutual information estimators and their associated parameters.

Families of Mutual Information Curves

The parameters of the mutual information estimators are the same as those for the entropy estimator, but each parameter is present for multiple, L, signals. For this investigation, the focus will be primarily on two signals and it is further assumed that the number of samples and kernel size is equal for both signals. The following list summarizes the variables to be explored for mutual information estimators:

1. N - The number of data samples or exemplars.

2. σ - The kernel size for the Parzen PDF estimate.
3. M_1, M_2 - The dimensions of the signals.
4. d_1, d_2 – A measure of the extent (or variance) of the signals.
5. L – The number of signals (only investigating L=2).

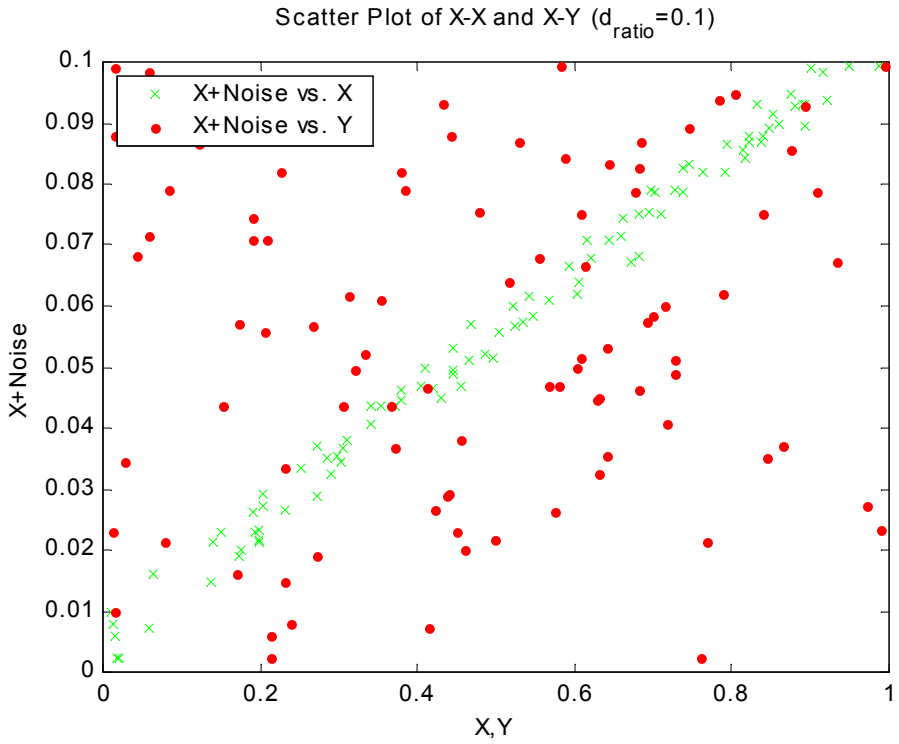

Figure 7-14. Scatter plot for mutual information experimentation data.

In order to make comparisons with the mutual information estimators, three test datasets are introduced. The first two datasets, ***X*** and ***Y***, are two independent N-sample datasets based on a random uniform distribution in M dimensions. The third dataset, ***Xn***, is the ***X*** dataset with additive Gaussian noise. These datasets are scaled by the ratio, $d_{ratio}=d_1/d_2$, in order to explore the effects of different data scales. These datasets are shown graphically in the scatter plot of Figure 7-14 for M=1 and d_{ratio}=0.1. The figure shows the expected correlation between ***X*** and ***Xn*** and the relative independence of ***Y*** and ***Xn***.

Effects of Sample Size Variations

The first set of curves, in Figures 7-15 and 7-16, explores the effects of the sample size, N, on self mutual information, $I(\boldsymbol{X},\boldsymbol{X})$. Notice first that $I(\boldsymbol{X},\boldsymbol{X})$ approaches zero for both CS-QMI and ED-QMI as the kernel size increases to large values. As the kernel size decreases, CS-QMI approaches a saturating limit defined by (7-21) as,

$$I_{CS\max}(L,N) = \lim_{\sigma \to 0} I_{CS} = (L-1)\log N \tag{7-21}$$

for the case where all N samples are unique. This saturating limit is shown in Figure 7-15 with the blue line. If some of the data points are nonunique, or repeated, a more complex expression results as described by (7-22).

$$I_{CS\max}(L,N) = (L-1)\log \frac{\left(N + \sum_{non-unique-sets} n_i!\right)\prod_{l=1}^{L}\left(N + \sum_{non-unique} n_i!\right)}{\sum_{j=1}^{N}\prod_{l=1}^{L} n_{j,l}} \tag{7-22}$$

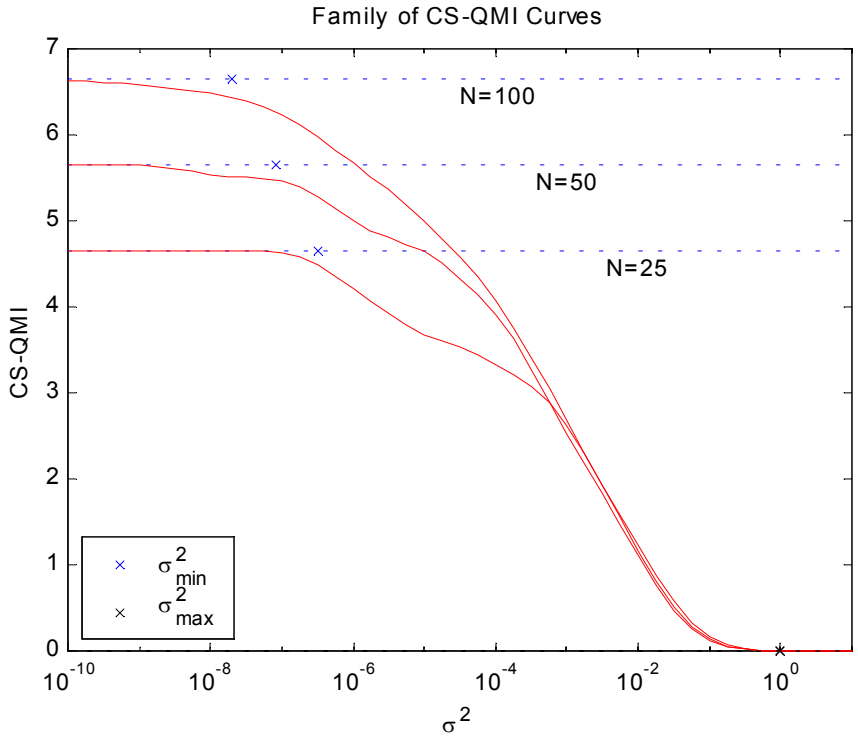

Figure 7-15. Family of CS-QMI curves: Variable sample size.

In (7-22) the first summation in the numerator is the sum of the factorials of the number of occurrences for each repeated set (or pair for L=2) of points in the dataset. The second summation is the sum of the factorials of each repeated point for each individual signal. The denominator sums the product of the number of occurrences of the j^{th} data point in each of the signals.

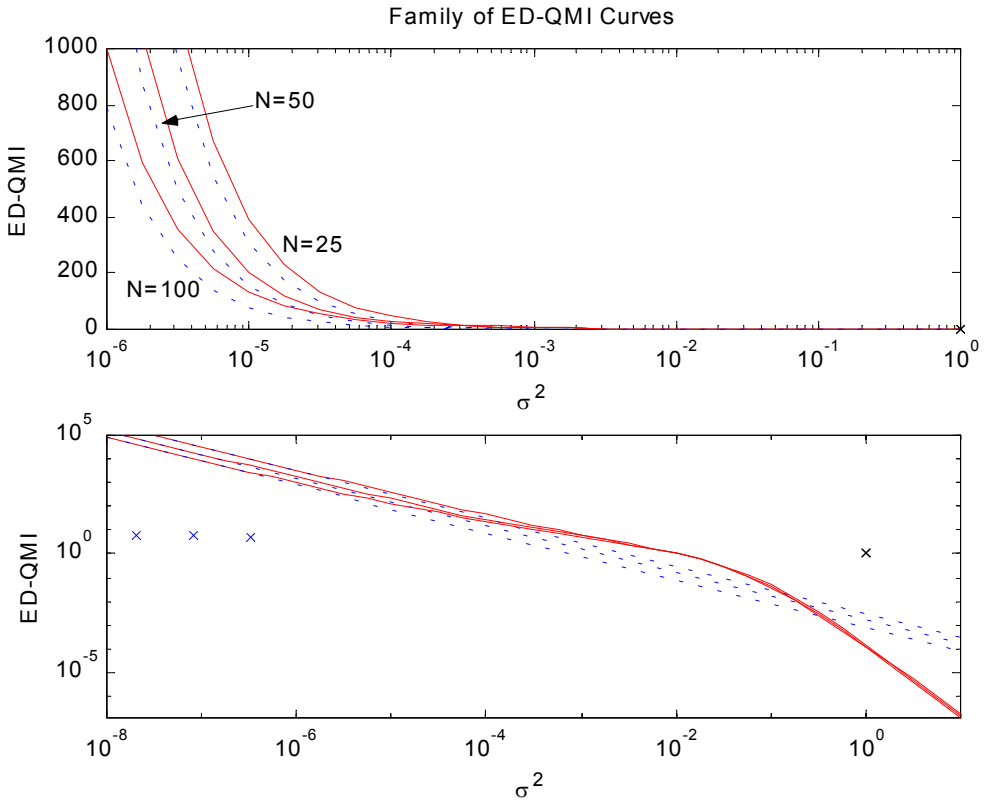

Figure 7-16. Family of ED-QMI curves: Variable sample size.

In the ED-QMI case, as the kernel size decreases, the estimator diverges because of the kernel size term in the denominator of the Gaussian kernels. Figure 7-16 shows both linear and logarithmic plots of the ED-QMI. The ED-QMI estimator approaches infinity exponentially as described in (7-23) for N unique samples and shown in Figure 7-16 with the blue lines.

$$I_{ED\max}(L,M,N,\sigma) = \lim_{\sigma \to 0} I_{ED} = \frac{1}{N\left(4\pi\sigma^2\right)^{LM/2}}\left(1-\frac{1}{N^{L-1}}\right) \tag{7-23}$$

Note that in the ED-QMI case, for small kernel sizes, the estimator approaches (7-23) from the top. For nonunique data samples, (7-24) describes the asymptotes more accurately where the summations and products are the same as in (7-22).

$$I_{ED\max} = \frac{\frac{1}{N}\left(N + \sum_{non-unique-sets} n_i!\right) + \frac{1}{N^L}\prod_{l=1}^{L}\left(N + \sum_{non-unique} n_i!\right) - 2\frac{1}{N^L}\sum_{j=1}^{N}\prod_{l=1}^{L} n_{j,l}}{N\left(4\pi\sigma^2\right)^{LM/2}} \tag{7-24}$$

In these figures, the blue and black crosses represent the kernel size extremes described by (7-5) and (7-6) for the entropy estimate. Since the components that make up the QMI estimates are essentially based on the Renyi's quadratic entropy estimator, these same kernel sizes appear to track the saturation of the QMI estimates as well. The plots in Figure 7-17 support this inference further. This figure shows the average information forces as a function of kernel size for the CS-QMI in red and the ED-QMI in green. Notice that for both cases, the information forces roll off in close relation to the kernel size extremes from (7-5) and (7-6).

As with the entropy estimators, comparison of mutual information estimates with different sample sizes can be achieved by utilizing the expressions in (7-21) through (7-24) to develop relative mutual information metrics as,

$$I_{CSrelative}(N_n) = \frac{I_{CS}(N_n, \sigma_n)}{I_{CS\max}(N_n)} \tag{7-25}$$

$$I_{EDrelative}(N_n) = \frac{I_{ED\max}(N_n, \sigma_n)}{I_{ED}(N_n, \sigma_n)} \tag{7-26}$$

where the kernel size, σ_n, should be adjusted as a function of N_n per the relationship in (7-17) or (7-6). Note also that (7-26) is only appropriate for small kernel sizes.

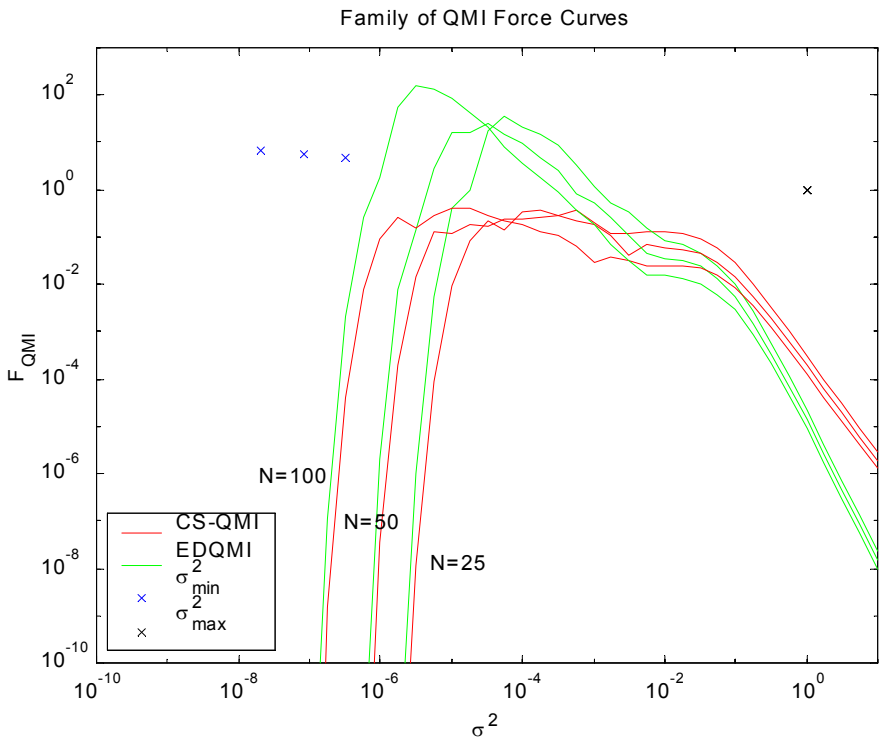

Figure 7-17. Family of QMI information force curves: Variable sample size.

Effects of Different Scale

This section examines the dependency of the mutual information estimators on the relative variance of the random variables that are being compared. Figures 7-18 and 7-19 show the mutual information between $\boldsymbol{X}$ and $\boldsymbol{X}$, denoted $I(\boldsymbol{X},\boldsymbol{X})$, in green and between $\boldsymbol{X}$ and $\boldsymbol{Y}$, denoted $I(\boldsymbol{X},\boldsymbol{Y})$, in red for various values of d_{ratio} where N=100 and M=1. Note here that it is expected that the self-mutual information, $I(\boldsymbol{X},\boldsymbol{X})$, should result in the maximum mutual information for a given set of parameters. Clearly for the CS case, the mutual information estimate shifts from left to right as the d_{ratio} increases. Although this behavior is undesirable, it is more troubling that for some kernel sizes even when d_{ratio}=1, the mutual information between the uncorrelated variables, $I(\boldsymbol{X},\boldsymbol{Y})$, is greater than $I(\boldsymbol{X},\boldsymbol{X})$. To allay the first problem, a reasonable approach is to normalize the variance of the two variables before comparing mutual information. The second problem

hints to the proper selection of kernel size. Clearly, when the kernel size is too small, the CS-QMI can produce unexpected results.

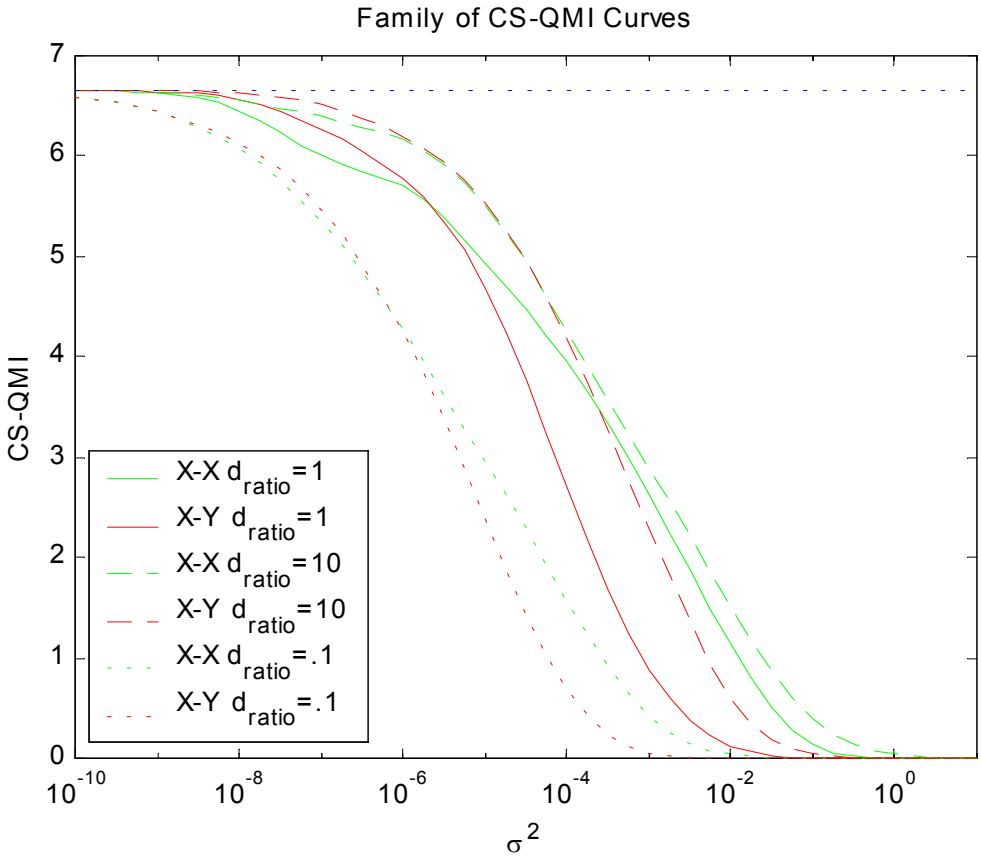

Figure 7-18. Family of CS-QMI curves: Variable scale ratios.

For the ED-QMI case, an increased d_{ratio} pushes out the roll-off on both ends of the kernel size scale and makes the difference between $I(\boldsymbol{X},\boldsymbol{X})$ and $I(\boldsymbol{X},\boldsymbol{Y})$ less discernable. Decreasing the d_{ratio} hastens the roll-off on both ends and increases the separation between $I(\boldsymbol{X},\boldsymbol{X})$ and $I(\boldsymbol{X},\boldsymbol{Y})$. Note that the ED-QMI is better behaved than CS-QMI in terms of the expectation that $I(\boldsymbol{X},\boldsymbol{X})$ be greater than $I(\boldsymbol{X},\boldsymbol{Y})$. The third frame in Figure 7-19 shows the difference between $I(\boldsymbol{X},\boldsymbol{X})$ and $I(\boldsymbol{X},\boldsymbol{Y})$ with positive values in blue and negative values in red. Note that the results are as expected for a wider range of kernel sizes than with the CS-QMI in Figure 7-18.

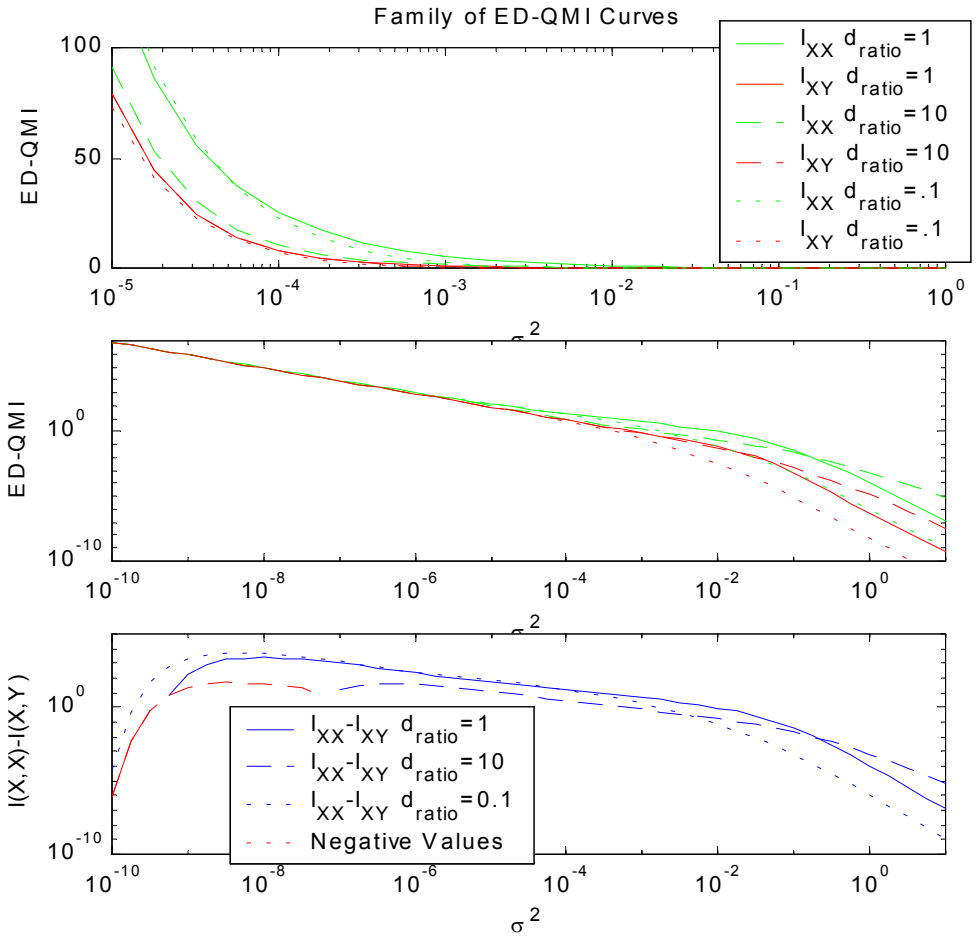

Figure 7-19. Family of ED-QMI curves: Variable scale ratios.

Effects of Dimensional Differences

Figures 7-20 and 7-21 show the effects of increasing dimension on the mutual information estimators. In these figures, the dimensionality of both variables is varied with M_1=M_2 for values of 1, 2, and 4. For these curves, $\boldsymbol{X}$ and $\boldsymbol{Y}$ are independent M-dimensional random uniform datasets consisting of N=50 samples. As with the entropy estimator, the slope of the transition increases with increasing M. Note here that the blue and black crosses denoting the kernel size limits from (7-5) and (7-6) seem to track the saturation points well with increasing M. Again, note the unexpected performance of the CS-QMI estimator for small kernel sizes.

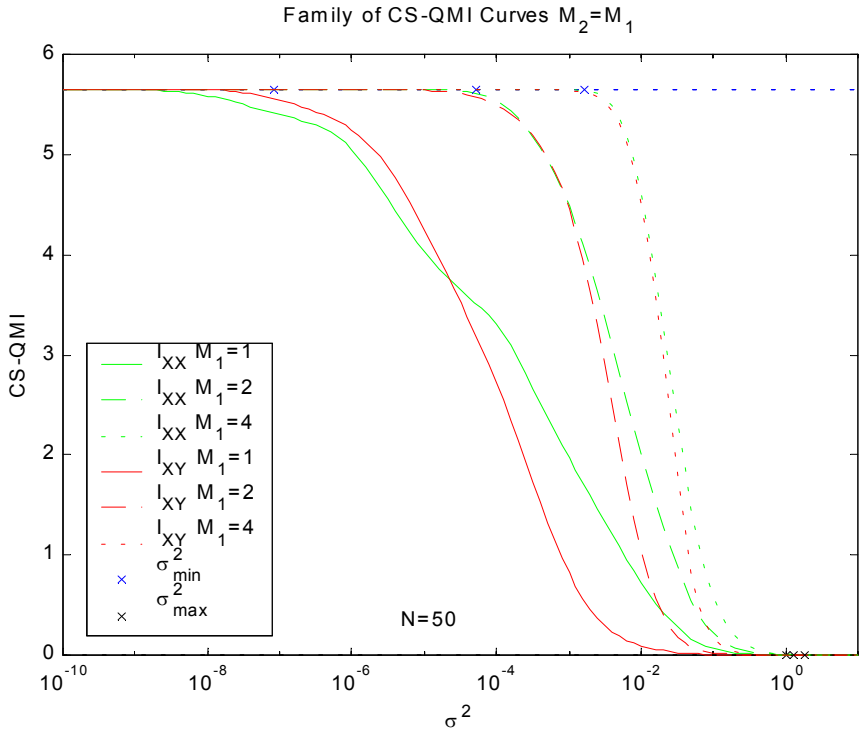

Figure 7-20. Family of CS-QMI curves: Variable equal dimensionality.

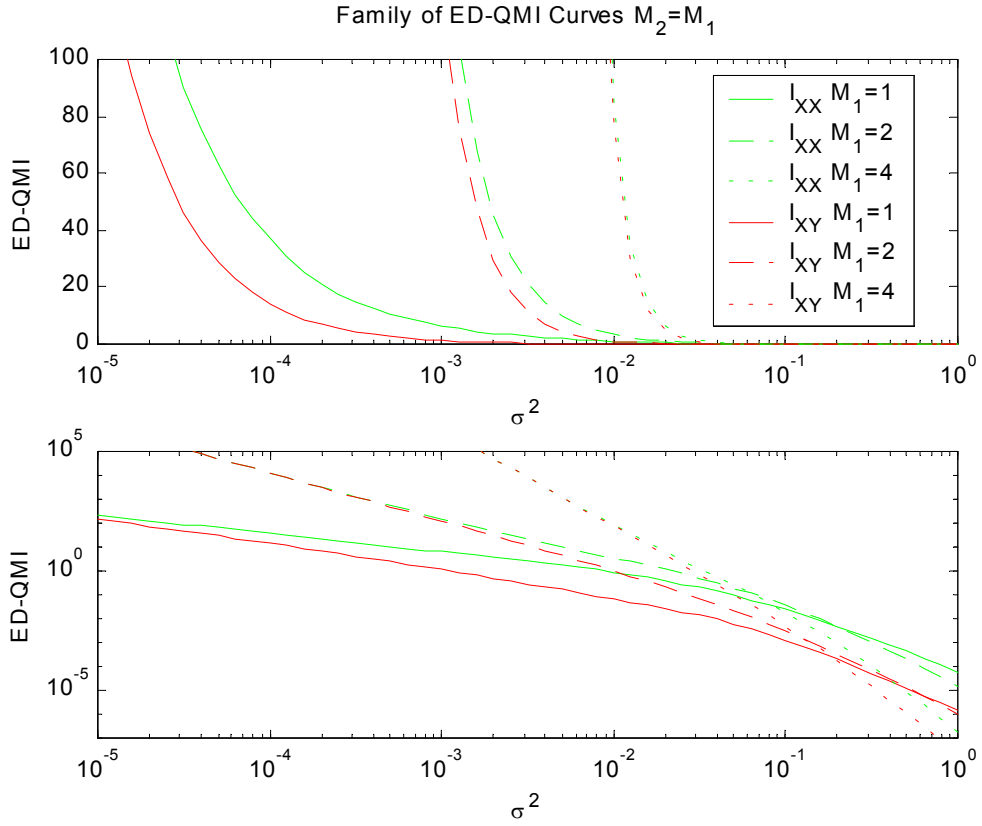

Figure 7-21. Family of ED-QMI curves: Variable equal dimensionality.

In Figures 7-22 and 7-23, the effects of increasing dimension on the mutual information estimators are explored when only one variable increases in dimension. This case represents an important class of problems where the goal is to determine which set of features with differing dimensions most closely represents a given target variable. In these curves, the dimensionality of M_1 is held fixed at one while M_2 varies from 1 to 4. Here, one variable, either $\boldsymbol{X}$ or $\boldsymbol{Y}$, is an M_2-dimensional random uniform dataset consisting of N=50 samples. The other variable, $\boldsymbol{X}_1$, is equal to the first dimension of $\boldsymbol{X}$ with additive Gaussian noise. So it is expected that the $I(\boldsymbol{X}_1,\boldsymbol{X})$ values should be greater than the values for $I(\boldsymbol{X}_1,\boldsymbol{Y})$. Note that the slope of the transition does not increase with dimension as in the previous case and that the kernel size ranges are perhaps modeled best as a function of either the maximum or minimum dimension using the expressions in (7-5) and (7-6) follows:

$$\sigma^2_{\max} = \sigma^2_{\max}(\max(M_1, M_2)) \tag{7-27}$$

$$\sigma^2_{\min} = \sigma^2_{\min}(\min(M_1, M_2)) \tag{7-28}$$

Note also that the mutual information estimate is generally higher with increasing dimensionality.

In order to be helpful in solving the stated problem, it is desirable for the smallest set that accurately represents the target set, $\boldsymbol{X}_1$, to have the largest value of QMI. At a minimum, it is desired for the sets that represent the target set to be greater than those sets that are uncorrelated regardless of dimensionality. Unfortunately, even in this simple case, none of the metrics performs ideally. First, note that the CS-QMI is again plagued with inverted results for small kernel sizes. For large kernel sizes, all the CS-QMI curves are greater for $I(\boldsymbol{X}_1,\boldsymbol{X})$ than for $I(\boldsymbol{X}_1,\boldsymbol{Y})$. However, the largest value corresponds to the

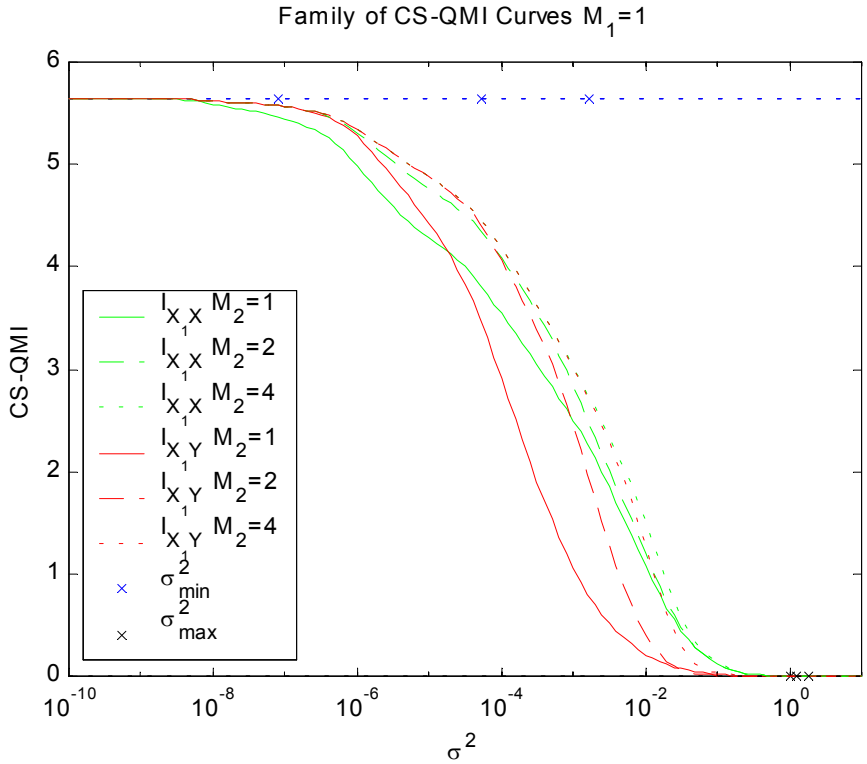

Figure 7-22. Family of CS-QMI curves: Variable different dimensionality.

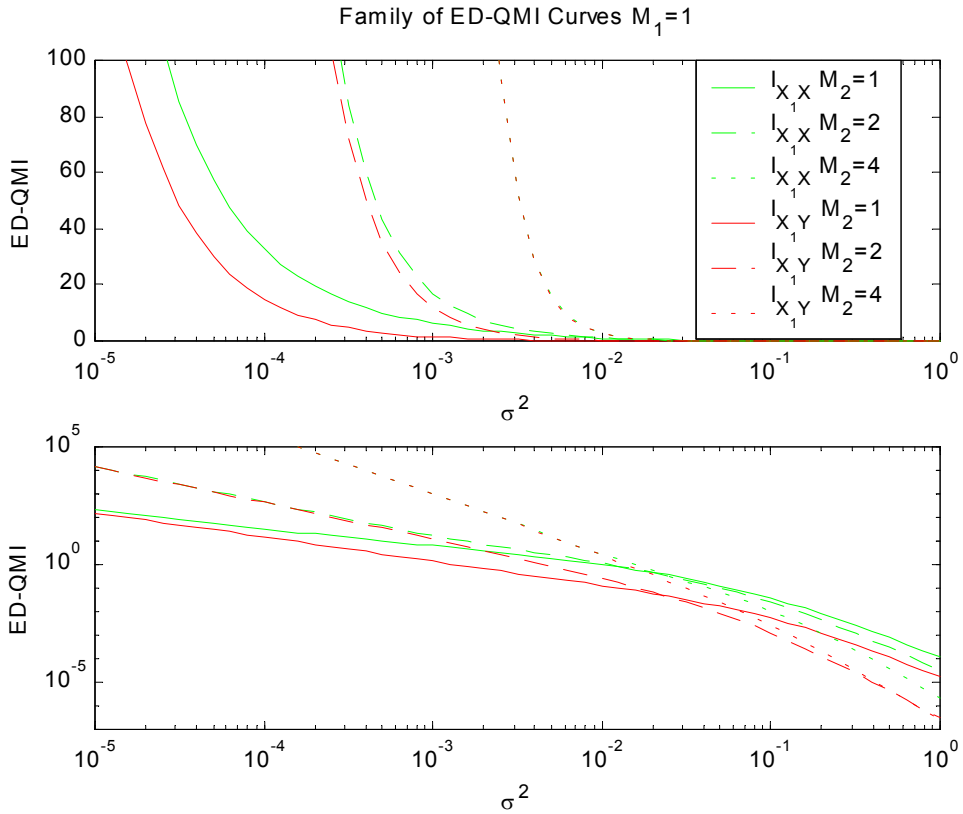

Figure 7-23. Family of ED-QMI curves: Variable different dimensionality.

largest dimension rather than the smallest as would be ideal. For ED-QMI, in each individual dimension, the $I(\boldsymbol{X}_1,\boldsymbol{X})$ is typically greater than $I(\boldsymbol{X}_1,\boldsymbol{Y})$ for most kernel sizes. However, comparisons between dimensions yield cases where uncorrelated sets of larger dimension are greater than correlated sets at small kernel sizes; and uncorrelated sets of smaller dimension are greater than correlated sets at large kernel sizes.

These difficulties have hindered the general application of the QMI estimators for feature relevance ranking. One possible remedy to compensate for the different dimensionalities is to select a largest common dimension in which to conduct all comparisons. Feature vectors in lower dimensions can be embedded in the higher dimension by simply repeating the information in the lower dimensional feature until the size of the largest common dimension is reached. Using this technique, the experiment from above is repeated where M_1 is held fixed at one, M_2 is held fixed at the largest common dimension, which is four, and the intrinsic dimensionality of M_2 is varied from one to four.

Figure 7-24 and 7-25 show the results for the CS-QMI and ED-QMI cases respectively. Notice that for the CS-QMI, the range in which the values for $I(\boldsymbol{X}_1,\boldsymbol{X})$ are greater than the those for $I(\boldsymbol{X}_1,\boldsymbol{Y})$ has increased. Again, for large kernel sizes, all the CS-QMI curves are greater for $I(\boldsymbol{X}_1,\boldsymbol{X})$ than for $I(\boldsymbol{X}_1,\boldsymbol{Y})$. However, in this case, the largest value corresponds to the smallest intrinsic dimension rather as is desired. For the ED-QMI the results have improved as well since $I(\boldsymbol{X}_1,\boldsymbol{X})$ is typically greater than $I(\boldsymbol{X}_1,\boldsymbol{Y})$ for most kernel sizes and, for most kernel sizes, comparisons between dimensions yield the expected results.

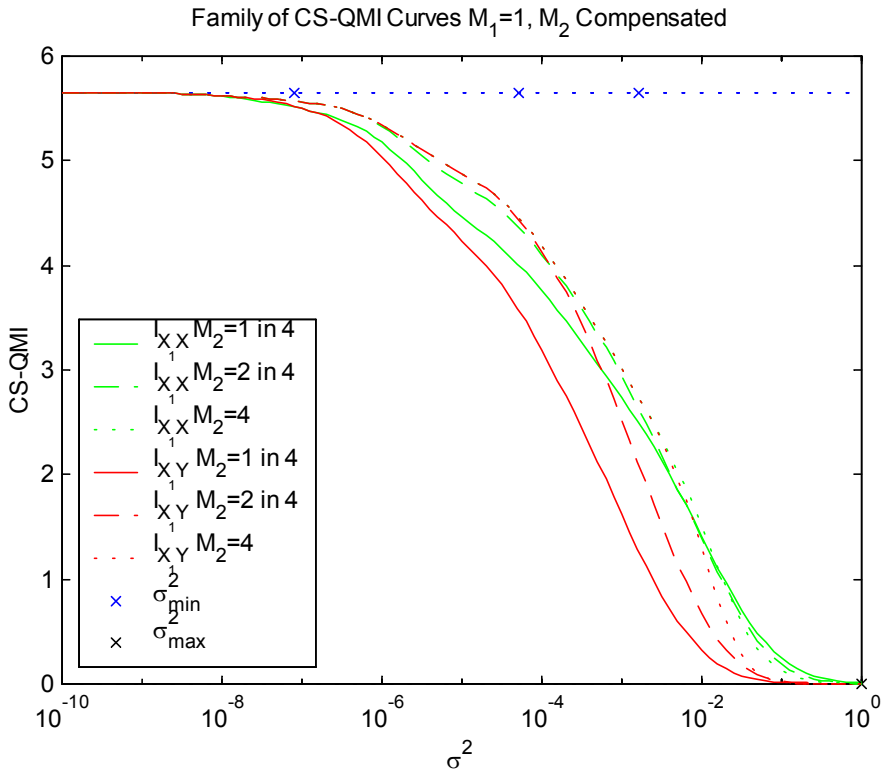

Figure 7-24. Family of CS-QMI curves: Variable dimensionality compensated.

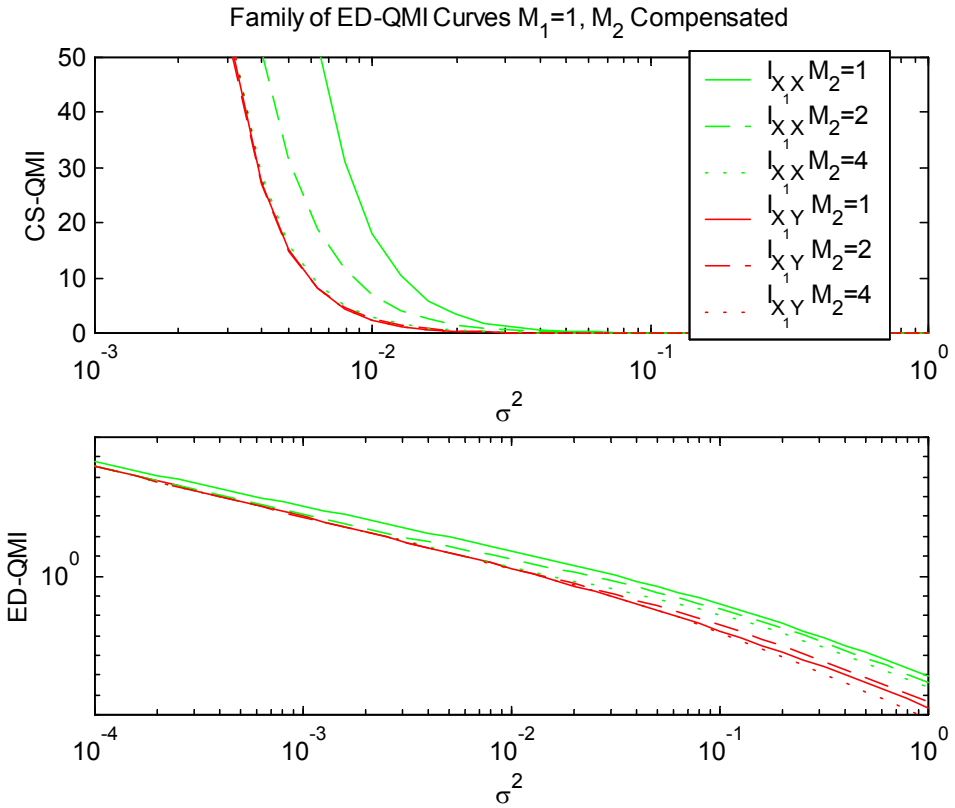

Figure 7-25. Family of ED-QMI curves: Variable dimensionality compensated.

Conclusions

In this chapter, the ITL estimators for Renyi's quadratic entropy and quadratic mutual information were studied by using a family of curves approach. These curves have provided insight as to the dependencies of the information estimators on the parameters of both the problem and the estimator. By exploring the effects of the kernel size, analytical expressions for the relevant range of kernel size were presented to avoid the effects of saturation. The consequences of comparing entropies estimated from variables with different scale were explored resulting in a scale-invariant form of the entropy estimator. Analysis of the effects of variable number of samples for entropy estimation resulted in the development of a relative entropy metric. Relative entropy offers a normalized estimate of entropy that appears useful for the determination of batch sizes in batch training implementation. Through the exploration of the effects of different dimensional data, two novel methods for the estimation of dimensionality based on information-theoretic measures were proposed. The first method, which is based on information potential, bears a close resemblance to the Grassberger-Procaccia method but seems to lack practical utility because of its complexity. The second method, based on relative entropy, appears promising for dimensionality estimation of sparsely sampled high dimensional datasets.

In the case of the quadratic mutual information estimators, the differences between and difficulties with both the CS-QMI and ED-QMI estimators were explored. Although hindered by their increased complexity, several useful insights as to the relationships as to the effects of parameter settings have been revealed. As with the entropy estimator, the saturating effects of kernel size ranges were explored. It was further shown that the mutual information estimators are even more sensitive to kernel

size often producing unexpected results for small kernel sizes. Analysis of the mutual information curves also resulted in a functional description of asymptotic behavior of the estimators for large and small kernel sizes. In addition, the impact of different scales of the information variables was explored, suggesting a normalization procedure to maintain consistent comparisons. Finally, the effects of varying dimension on the mutual information estimators were examined. In order to compensate for the strictly increasing effects of increased dimension, a method is proposed to embed lower dimensional data in a common higher dimension by using redundant information.

The properties and insights described herein, although not exhaustive or completely rigorous, serve as a foundation upon which a better general understanding of the information estimators involved in the ITL process can be achieved. By elucidating the properties of the estimators and the relationships between the parameters and the resulting information estimates, the application of information-theoretic principles can be shared by a much wider audience.

CHAPTER 8
SONAR FEATURE EXTRACTION

Introduction

In this chapter, the concepts of information-theoretic learning, supplemented with the algorithmic and training enhancements presented heretofore, are applied to the feature extraction stage of sonar ATR. In general, the feature extraction process in ATR is challenging problem for a variety of reasons. In many cases, the number of exemplars available for training is small in relation to the dimensionality of the data. Historically, this has been the case for the sonar ATR problem because of the expensive cost of real data collection. This mismatch brings about the effects of Bellman's curse of dimensionality discussed in Chapter 2, which stresses the importance of dimension reduction in order to promote generalization that is more robust.

This chapter presents several methods for extracting and selecting feature sets for sonar ATR based on information-theoretic principles with the goal of dimension reduction. The methods were developed and tested using the SON10 and SON512 feature sets described in Chapter 2. First, a maximum entropy method is proposed for informative feature extraction and compared to its PCA counterpart. Next, mutual information is used to guide an information-theoretic approach to feature subset selection. Lastly, mutual information is used to extract new discriminative feature sets of greatly reduced dimensionality from the original feature set. The results of these methods are compared with some of the most common and current methods used in practice today in terms of both training set performance and test set generalization. Although the methods

described are applied using existing extracted feature sets as the input space, they can readily be applied to the raw or preprocessed image data as well.

Informative Feature Mapping

Informative feature extraction is important in the cases where the available data is sparse relative to the dimensionality of the data. Dimension reduction is desired in order to bring the dimensionality of the feature space in line with the available training data so that the training process will yield acceptable generalization results. In this case, the goal is to generate a new set of features from a given set of data or features that reduces dimensionality and, hopefully, compresses relevant class information. The new set of features is typically derived as some linear or nonlinear combination of the original features. Here it is desired to extract features based on the information content of the original feature space without applying any class information to bias the mapping. Perhaps the most common method for informative feature extraction is PCA. In this section, a method is proposed for informative feature extraction using maximum entropy as the criterion.

Approach

Given a final desired feature dimension of M_0, the goal of this approach is to extract features that compress as much information as possible into an M_0-dimensional space using the MaxEnt criterion. The motivation for this method is dimension reduction to allow for better generalization performance of the trained system. First, the MaxEnt criterion was applied using a linear combination of the original features. The resulting features performed roughly equivalent to PCA in the training set but slightly worse for test set generalization. This motivated the use of a nonlinear mapping in hopes of obtaining better performance though more efficient information compression. Initially,

experiments were made by applying the MaxEnt criterion to a MLP with 60 inputs, one for each original feature, a variable number of hidden layers, and M_0 output features with M_0 varying from one to ten. This resulted in a network with several hundred weights and was therefore difficult to train well because of the limited number of exemplars. In other words, the feature extraction network was suffering from Bellman's curse because the number of parameters in the network was too large in relation to the number of exemplars available to train it.

In order to prune down the size of the feature extraction network, an approach evolved that combines the linear dimension reduction of PCA with the nonlinear characteristics of a MaxEnt-trained MLP. Based on intrinsic dimensionality estimates of the original data ranging from approximately four to eight, it was clearly unnecessary to begin with the full dimensionality of 60. Given a final desired feature dimension of M_0, PCA is used for dimension reduction using the full dimensional feature set to reduce the dimension to $3M_0$. Since PCA is known to overestimate dimensionality, the factor of three was chosen to include additional information that might be compressed by a more efficient dimension reduction algorithm, such as MaxEnt, into the desired feature dimension. Thus, the resulting $3M_0$ PCA features are used as input into a $3M_0$-12-M_0 MLP with MaxEnt criterion.

In order to improve the training efficiency of the MLP, the batch training approach from Chapter 6 is used to divide the training set into smaller batches of 100 samples using a special batch selection method. Since there are a disproportionately larger number of false alarms than targets in these sonar datasets, the batch selection method uses the entire target set all the time with the remainder of the 100 samples

randomly selected from among the false alarms. The number of batches (NB) is 200 and the number of epochs per batch (NEB) is three. The SCG advanced training algorithm for ITL from Chapter 5 is used to search the parameter space of the MLP with a fixed kernel size of 0.1. The entire process is shown in Figure 8-1.

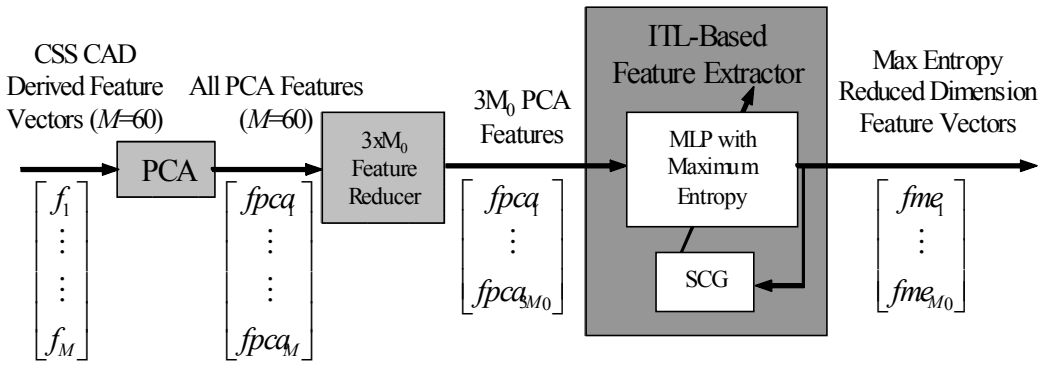

Figure 8-1. Maximum entropy informative feature extraction for dimension reduction.

Results

Since the goal of this approach is informative feature extraction, it is desired to compare which feature set retains the highest information content. Although this is not a straightforward comparison, Figures 8-2, 8-3, and 8-4 attempt to provide a basis for informative comparison in two ways. First, since the eventual goal is classification performance, it is desired to compare which features provide the best training and test set classification results. By generating ROC curves, it is possible to ascertain how much discriminative information is preserved. Keep in mind, however, that since there is no class information involved in the feature extraction process, this may not accurately represent how well the MaxEnt feature extraction process is actually working. A second method for comparing feature sets looks at information content without regard for class

discrimination information. By using the relative entropy metric from Chapter 7, it is possible to compare the relative entropies of PCA features with the MaxEnt features. In this case, the caveat is that although more information may be present, it may have nothing to do with the eventual goal of class discrimination.

Figure 8-2 shows the ROC curves for the PCA features (solid lines) and the MaxEnt features (dotted lines) for various values of M_0 using the SON10 feature dataset and a multivariate Gaussian (MVG) classifier. The curves in red represent training set classification performance while the blue curves represent test set generalization performance. Since the MaxEnt features were generated using a randomly initialized MLP, an average of 30 runs for each dimension was used to create the MaxEnt ROC values. Note that for M_0=1, PCA and MaxEnt perform quite similarly with PCA performing slightly better overall. For M_0={2,3,4,5}, however, the MaxEnt features are clearly superior in both training set performance and test set generalization. Then, for M_0=6, the performance of the features sets is again quite similar. Figure 8-3 shows similar results for the SON512 dataset.

It was conjectured that the reason for these similarities and differences in performance could be attributed as follows. First, for M_0=1, the first PCA feature is either quite efficient in information compression or, by coincidence, contains a significant amount of classification information. For the middle ranges of M_0 in Figure 8-2, the MaxEnt criterion is able to compress more information into these smaller dimensional spaces than PCA. Some of this additional information, again coincidentally, bears additional discriminative information as well. As M_0 increases to larger values, two factors come into play. First, the intrinsic dimensionality of the dataset is approached, so

there is little missing information left from the perspective of PCA. Second, as M_0 gets larger, the complexity of the feature extraction network increases and the mismatch between the number of exemplars and the number of free parameters comes into play again.

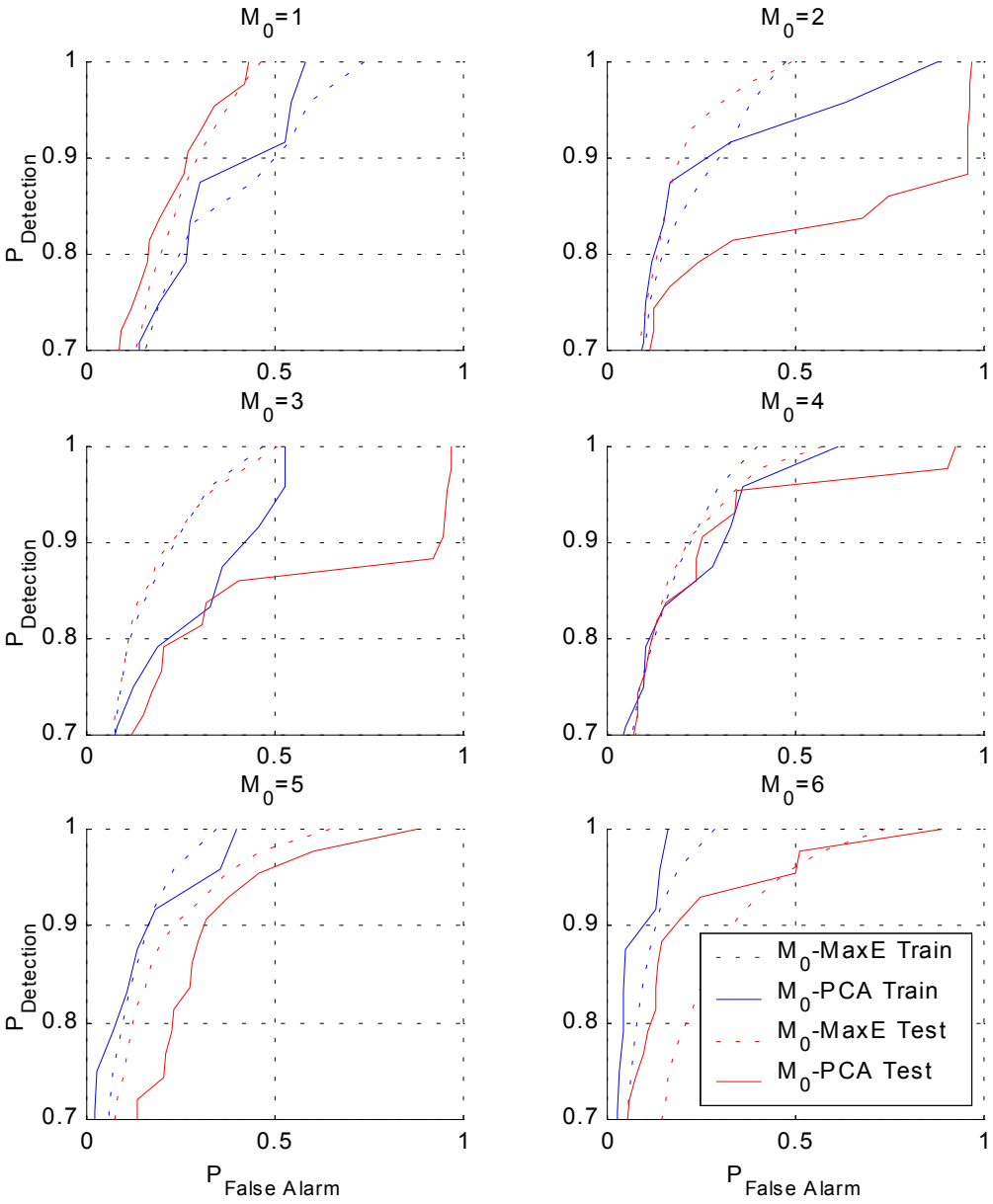

Figure 8-2. Classification performance of MaxEnt and PCA features for SON10.

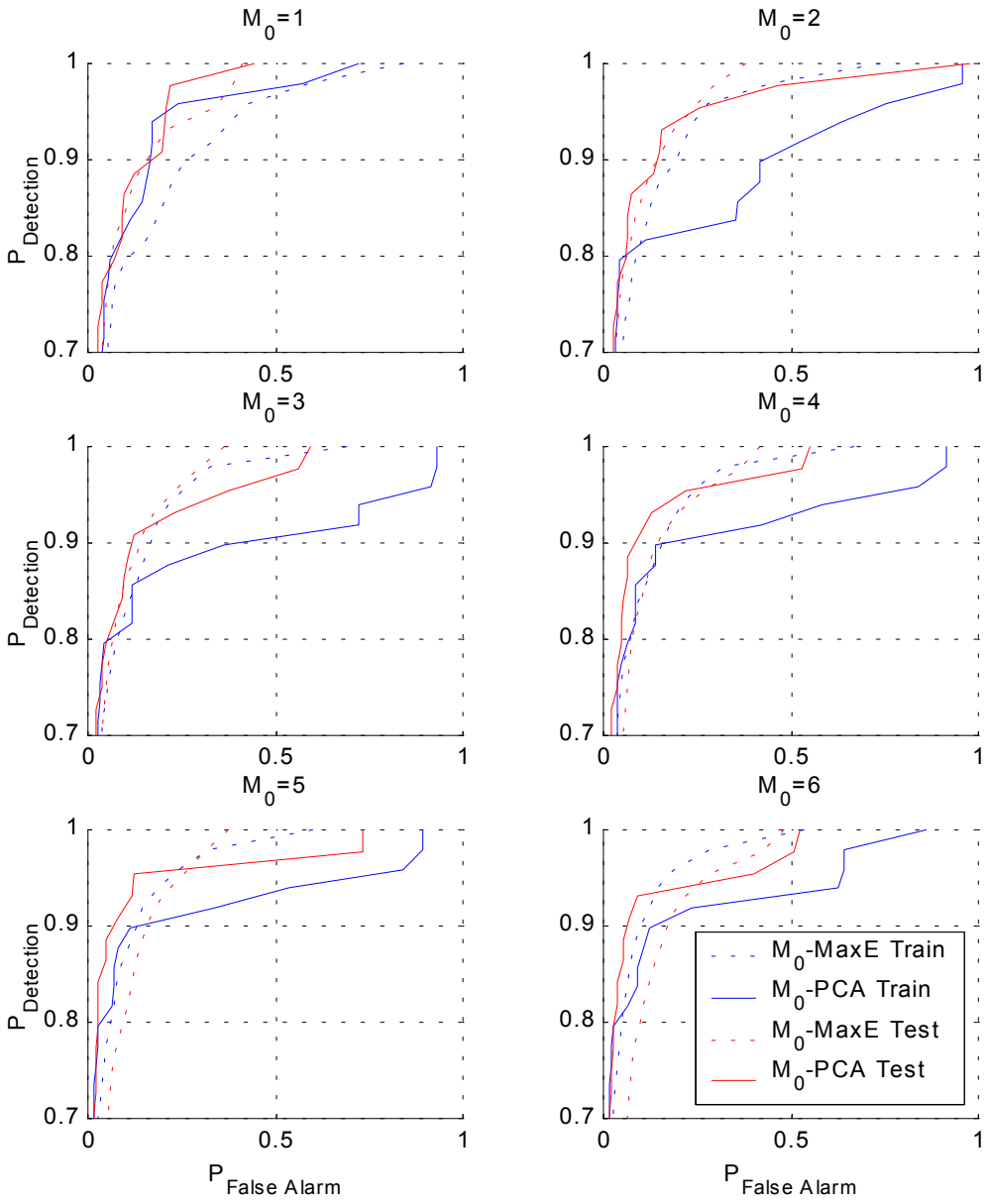

Figure 8-3. Classification performance of MaxEnt and PCA features for SON512.

In addition to providing another metric for comparison of the informative feature extraction method, the relative entropy measure from (7-18) can also be helpful in exploring some of the above conjectures. Figure 8-4 shows the relative entropy versus the feature dimensionality, M_0, for the PCA features in blue and the average for the MaxEnt features in red. The green lines show the maximum and minimum relative

entropies from among the 30 MaxEnt trials. First, note that the relative entropy tends to decrease with dimension; this is expected since the relative entropy measure from (7-18) has not been normalized with respect to dimensionality.

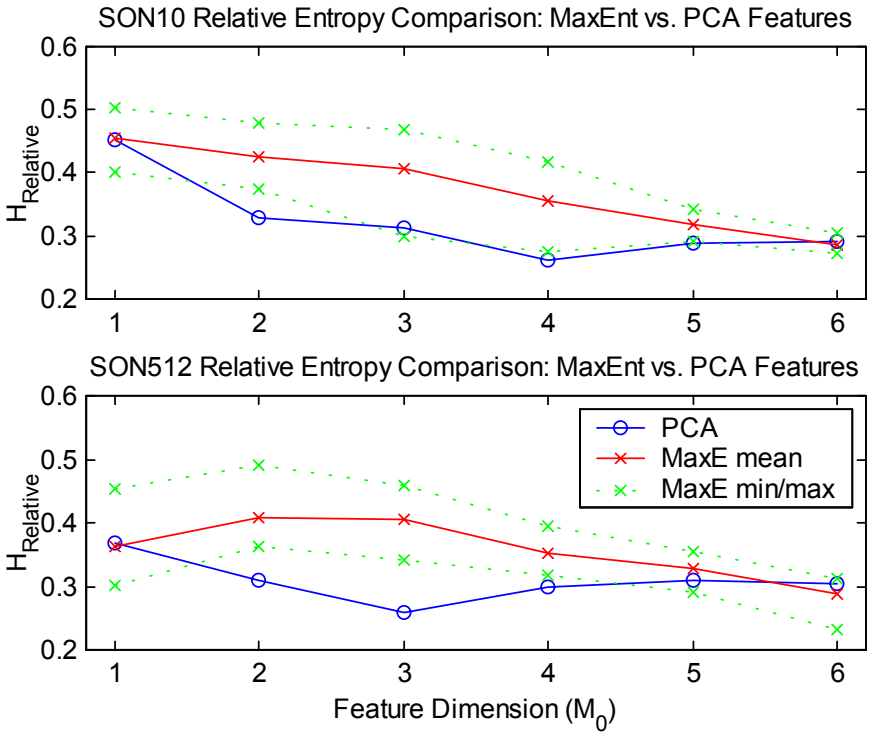

Figure 8-4. Comparison of relative entropy of MaxEnt and PCA features.

Notice that in the case where M_0=1, the PCA and MaxEnt features have approximately the same relative entropy. This indicates that both features contain roughly the same amount of information and, based on Figure 8-2, roughly the same amount of discriminative information. In the range of for M_0=2,3,4,5, the relative information for the MaxEnt features is greater than that for the PCA. This suggests that for this range in dimensionality, the MaxEnt criterion is able to compress additional information into these dimensions than PCA. Figures 8-2 and 8-3 corroborate this interpretation at least in terms of discriminative information content, although it is important to note that this might not generally be the case. As M_0 gets larger, the PCA

and MaxEnt relative information values converge once again. This suggests that there is little additional information beyond what the PCA features can represent in these dimensionalities.

Feature Subset Ranking and Selection

One of the more basic methods for feature extraction with the goal of dimension reduction is the feature subset ranking and selection approach. This method ranks subsets of the original feature set according to some figure of relevance and then selects from among the best-ranked subsets of original features. Some of the more common ranking criteria are the Bayes error and various distance measures. It should be noted that this approach suffers from a combinatorial number of possible candidate feature subsets, which renders an exhaustive search to be computationally prohibitive in most practical cases. Some procedures have been developed to avoid exhaustive search, such as forward and backward search; however, they cannot assure selection of the best subset. A branch and bound method has been developed that helps to reduce the demands of exhaustive search and guarantees an optimal solution for monotonic criteria, however, this method can also be computationally prohibitive for large dimensional spaces. In the sections below, a method for ranking features using a criterion based on the mutual information between the features and the class labels is presented.

Approach

This goal of this approach is to select from a set of existing features a subset that emphasizes the discriminatory information between the classes in the data using the ClassInfoFilt criterion. The motivation for this method is to select the minimum dimensional feature subset that still provides the desired performance in terms of class separability between targets and false alarms. By evaluating the mutual information

between candidate feature sets and the class labels, the candidate sets can be ranked in order of their mutual information. It is expected that the best-ranked feature sets will contain features that tend to preserve and emphasize the separability between classes so that subsequent processing (i.e., the classifier stage) will be able to partition the classes easily. There are a variety of ways to determine what features should be members of the subset including exhaustive search, forward selection, and backward elimination. Regardless, of the search method, the output of the procedure is a reduced dimensional space of M_0 original features suited for class discrimination. The entire process is shown in Figure 8-5.

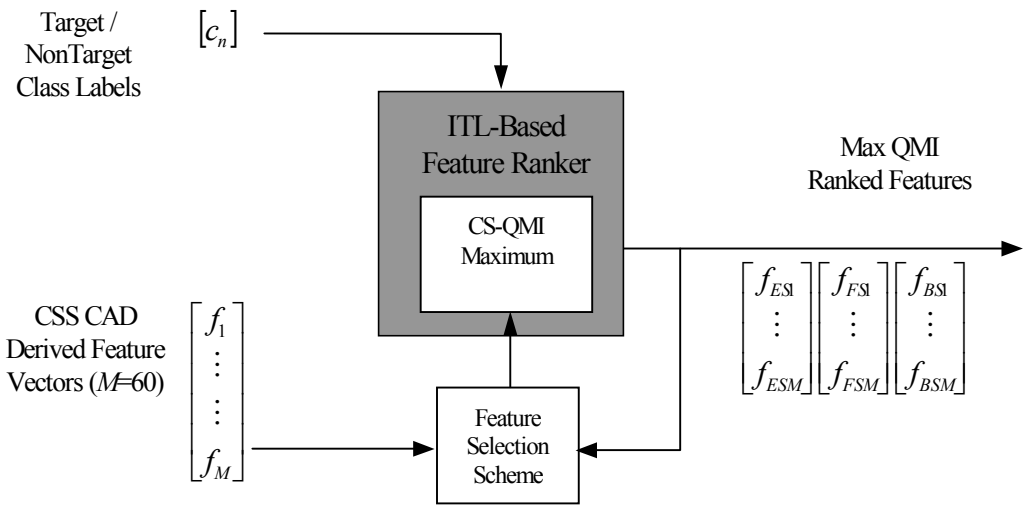

Figure 8-5. Maximum QMI feature ranker.

The exhaustive search approach is only feasible for small dimensions. It was performed for the SON10 and SON512 training datasets for dimensionalities up to four and three, respectively. Table 8-1 lists the optimum feature sets, based on CS-QMI, for these feature set sizes along with the time required to complete the exhaustive search. The table also includes the extrapolated time estimates required to complete the exhaustive searches for larger dimensionalities. Clearly, the combinatorial nature of

exhaustive search combined with the computational complexity of the ITL criterion severely limits the applicability of this method. Furthermore, even the results presented here would be computationally impractical without implementing the optimization results from Chapter 4.

Table 8-1. Exhaustive search feature sets and times for SON10 and SON512.

Dimension	Number of Combinations	SON10 Optimum Feature Set	SON512 Optimum Feature Set	Approx. Time to Compute
1	60	{10}	{37}	30 sec
2	1770	{7,10}	{37,8}	15 min
3	34,220	{7,8,10}	{8,9,12}	5.5 hrs
4	487,365	{3,7,8,10}	?	4 days
5	5,461,512	?	?	35 days (est.)
6	50,063,860	?	?	300 days (est.)

A simple method to explore higher dimensional subsets is forward feature selection. The first step in the forward selection process is to rank the features according to their individual relevance, which in this case is the CS-QMI. Figure 8-6 shows the mutual information estimate of each of the individual features with the class vector for both SON10 and SON512. The best individual feature, in these cases, feature #10 for SON10 and feature #37 for SON512, is used to start the forward selection feature list. Then, features are added one by one according to which feature increases the mutual information with the class vector by the largest amount. This provides an ordered ranking of the feature space for an arbitrary number of features. Using this approach, the features for SON10 and SON512 were computed. The best 15 in order were determined as described in (8-1) and (8-2).

$$f_{FS}(SON10) = \{10,7,8,3,9,61,12,11,46,13,56,59,58,1,16...\} \tag{8-1}$$

$$f_{FS}(SON512) = \{37,8,9,12,7,46,26,4,36,10,16,60,47,13,61,...\} \tag{8-2}$$

This method requires the evaluation of 1829 different feature sets and takes approximately 45 minutes to complete. Note that in the case of SON10, the forward selection method correctly selects the first four features as optimal when compared to the exhaustive search. Beyond that, however, there is no guarantee of optimality. For SON512, forward search only selects the optimum set up to dimension two, as feature #37 is not among the best three listed in Table 8-1.

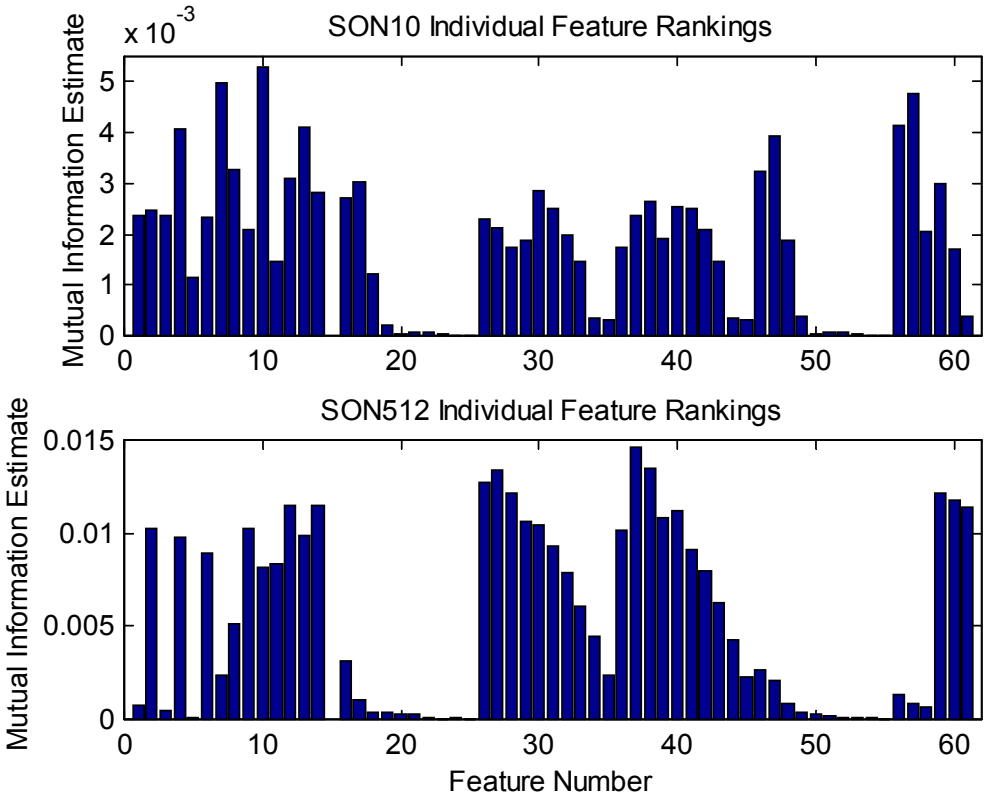

Figure 8-6. Individual feature CS-QMI for SON10 and SON512 training sets.

Backward elimination is another simple method that allows for a computationally tractable way of exploring higher dimensional subsets. To execute a backward search, the mutual information is computed using the entire original feature set. Then, features are removed one by one that reduce the mutual information by the smallest amount. This method also provides an ordered ranking of the feature space for an arbitrary number of

features. Using this approach, the features for SON10 and SON512 were computed with the best 15 listed in (8-3) and (8-4).

$$f_{BS}(SON10) = \{7,8,9,61,12,11,46,13,3,56,59,58,1,16,57...\} \quad (8\text{-}3)$$

$$f_{BS}(SON512) = \{26,8,16,7,36,46,4,10,9,37,12,61,47,60,11...\} \quad (8\text{-}4)$$

This method sets requires the evaluation of 1829 different feature sets and takes approximately 105 minutes to complete. Note here that for both data sets the backward search does not produce the optimal set. Feature #10 is eliminated midway through the elimination process for SON10, and features #9 and #12 are eliminated too early for SON512.

Another approach for exploring higher dimensional spaces is to combine the forward and backward search in repeated succession. This approach is similar to that used in the feature selection process of Dobeck [Dob97] with the figure of merit criterion replaced by the CS-QMI between the candidate feature set and the target class. However, the QMI criterion, as explained in Chapter 7, does not lend itself to this process because of the monotonic nature of the criterion. As noted in Chapter 7, the QMI estimators tend to increase as the dimensionality increases regardless of whether or not the new features contain additional relevant information or not. To counteract this effect, the compensation for dimensionality comparison suggested in Chapter 7 is applied to the feature selection process. Figure 8-7 shows the difficulty with the basic criterion in blue for both SON10 and SON512 datasets. It plots the mutual information estimate against the dimensionality of the feature set as features are successively added during the first forward search. Note that for the basic criterion, the mutual information estimate is strictly increasing. The red lines show the method from Chapter 7 where the

dimensionality is held constant at a fixed value, in this case 15, for all feature sets with redundant information used to fill the space for dimensions lower than 15. Note here that the feature sets reach a maximum mutual information level at dimensions in the range of 5 to 8.

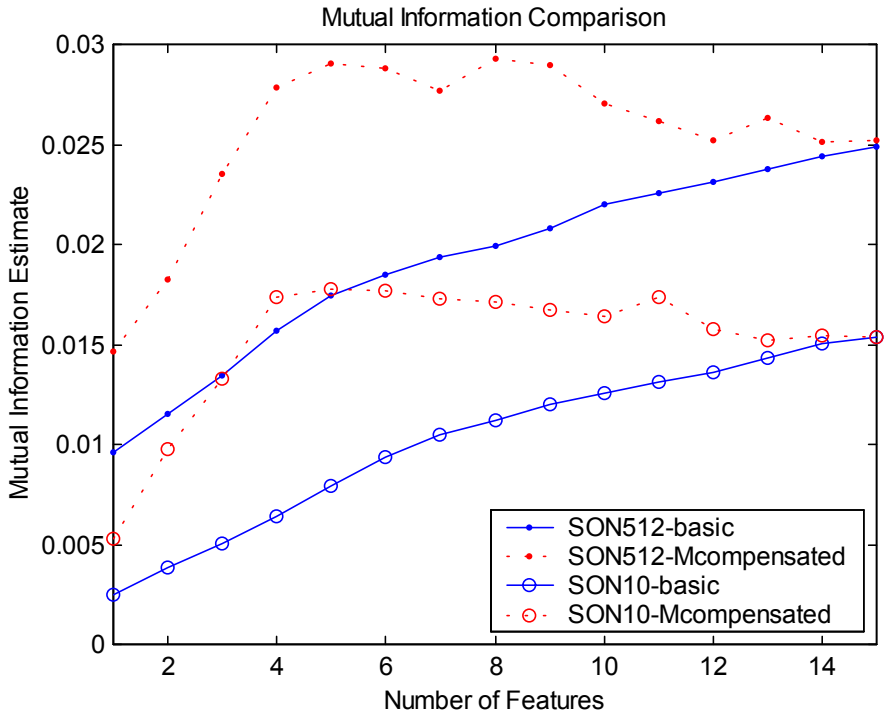

Figure 8-7. Mutual information estimates with and without dimension compensation.

This forward/backward search method was used to obtain an estimated optimum feature set for both the SON10 and SON512 datasets. Table 8-2 lists the estimated optimum feature sets for this method. These searches resulted in feature sets of dimension five and six, respectively. Note that the time required to compute these sets was only on the order of 15-20 minutes per dataset. Also, note that for both cases, the computed exhaustive search results are at least a subset of the estimated optimum using the combination search. Remember, however, that neither the forward, backward, nor combination methods can guarantee the optimal solution.

Table 8-2. Forward/backward search feature sets for SON10 and SON512.

Dataset	Combinations Evaluated	Optimum Feature Set	Time to Compute
SON10	809	{3,7,8,10,57}	15 min
SON512	1370	{7,8,9,12,16,36}	20 min

Results

In order to compare the downstream performance of the various selected feature sets, a MVG classifier is used to generate ROC curves for each of the cases. ROC curves are also generated for PCA features of various dimensions and for the features generated by the LDA method. Figures 8-8 through 8-13 shows the ROC performance for the QMI selected feature sets (dotted lines) of various dimensions compared to the PCA (solid lines) and LDA techniques (dashed lines). Figures 8-8 and 8-9 show the curves for the SON10 training and test sets, respectively, using the forward selection method for M_0=2,4,6,8,10, and the forward-backward search method that resulted in M_0=5. Figures 8-10 and 8-11 show the curves for SON10 training and test sets using the backward elimination. Lastly, Figures 8-12 and 8-13 show the curves for SON512 using the forward search and the forward-backward search process that resulted in M_0=6.

In Figure 8-8 it is evident that the QMI ranked features, for each dimension from two through ten, clearly outperform their PCA counterparts. They also provide better training performance than the LDA method for high values of $P_{Detection}$. The combination forward-backward feature set of dimension 5 provides performance in between the forward search features of dimension 4 and 6 as might be expected. In Figure 8-9, the test set generalization of the QMI ranked features again outperforms their PCA counterparts and the LDA method for high values of $P_{Detection}$. In this case, the forward-backward feature set provides the best generalization of all for high values of $P_{Detection}$.

Figures 8-10 and 8-11 show that the feature sets generated using backward elimination, while still outperforming their PCA counterparts in most cases, underperform their forward selection counterparts. While performing well against LDA in the training set, the backward elimination feature sets perform worse than the LDA features in test set generalization. The next two figures, 8-12 and 8-13, show that the results for the SON512 dataset were not as good for the QMI ranked features. Although the QMI ranked features performed better than PCA in most cases for high values of $P_{Detection}$, they performed worse that the LDA features in both the training and test set cases.

These results are generally as expected. By taking into account class information during the feature selection process, the QMI ranked features generally outperform the PCA features that do not have access to class information. When compared to LDA, which does have access to class information, the QMI ranked features performed better in one case (SON10) and worse in the other (SON512). The advantage the LDA and PCA methods have is that their features can draw from a linear combination of all 60 features, whereas the ranking method uses only a subset of the features with dimension M_0. In the SON512 case, this appears to be important. Overall, the QMI ranking method appears to be a viable method for guiding a feature ranking and selection process. In this light, it is a criterion that can be used to determine the relevance of existing features and any candidate new features that might be under consideration for the feature extraction stage. Note in Figure 8-6 that many of the features provide little to no class relevant information in the case of either the SON10 or SON512 datasets.

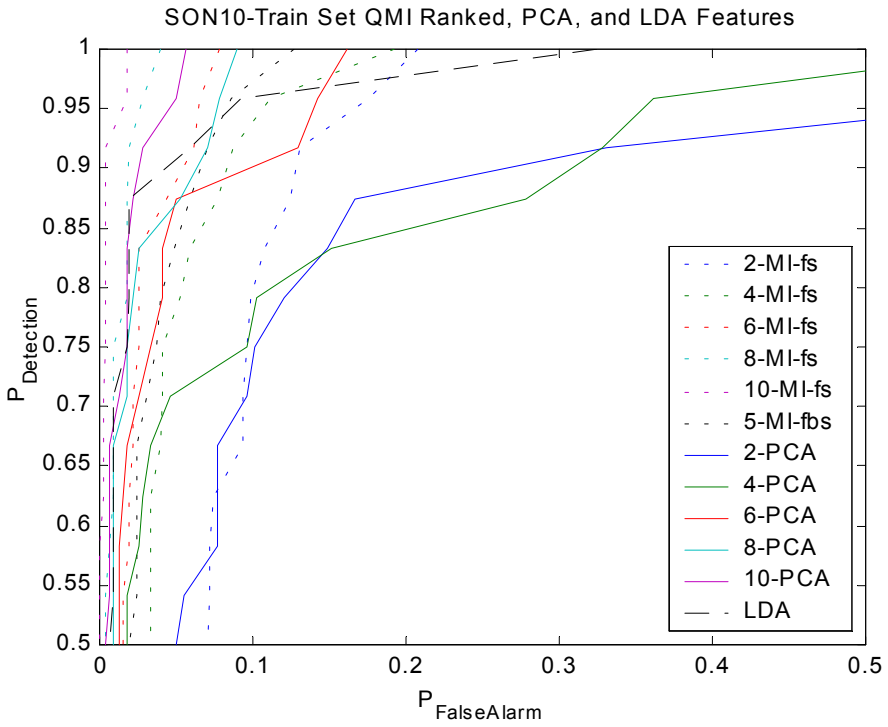

Figure 8-8. Classification performance of QMI ranked features with forward selection for SON10 training set.

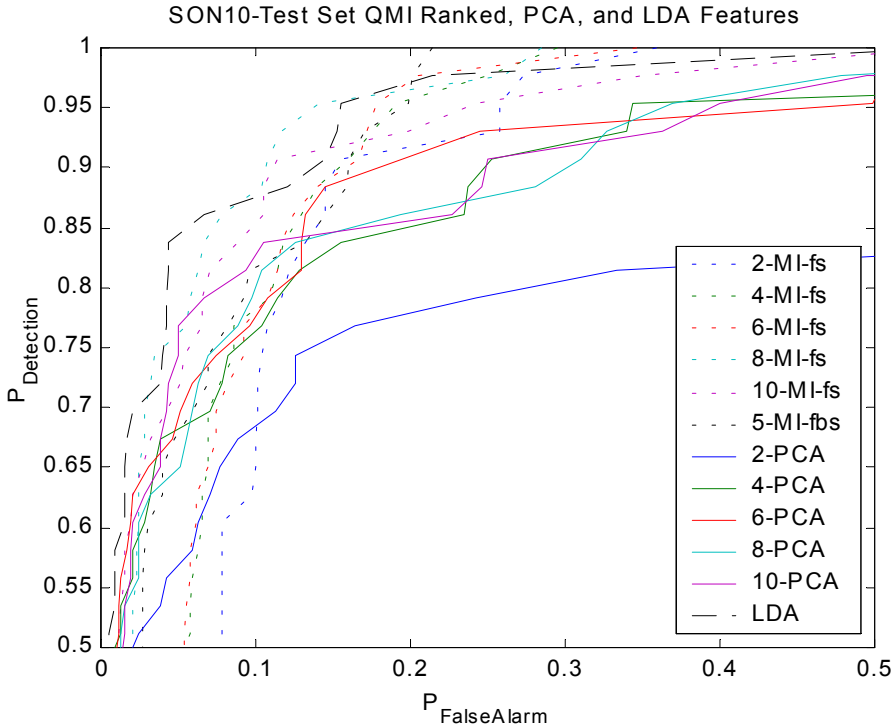

Figure 8-9. Classification performance of QMI ranked features with forward selection for SON10 test set.

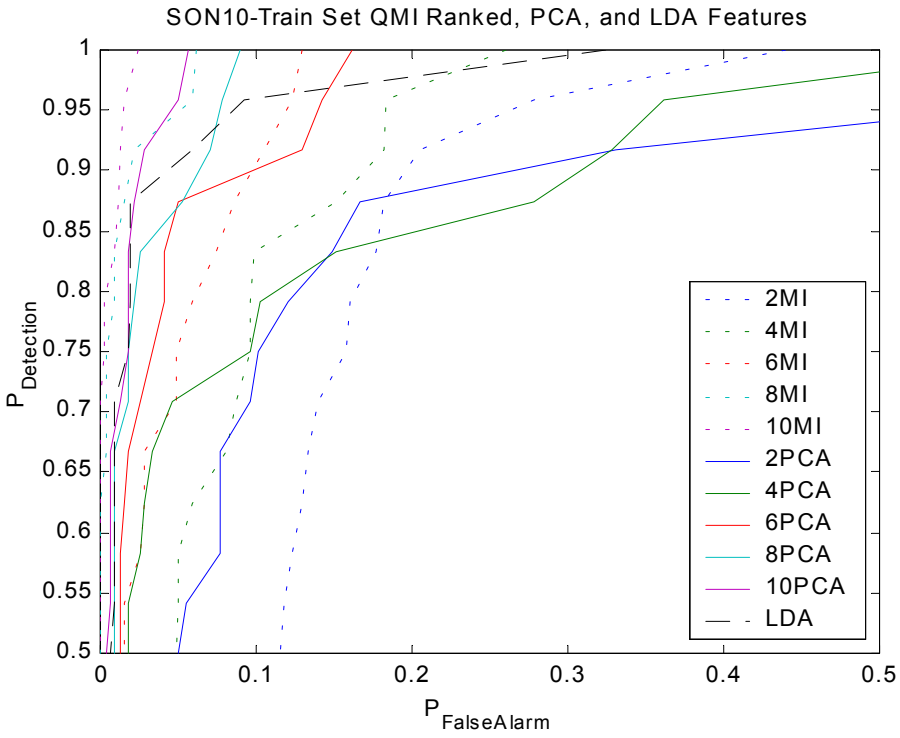

Figure 8-10. Classification performance of QMI ranked features with backward elimination for SON10 training set.

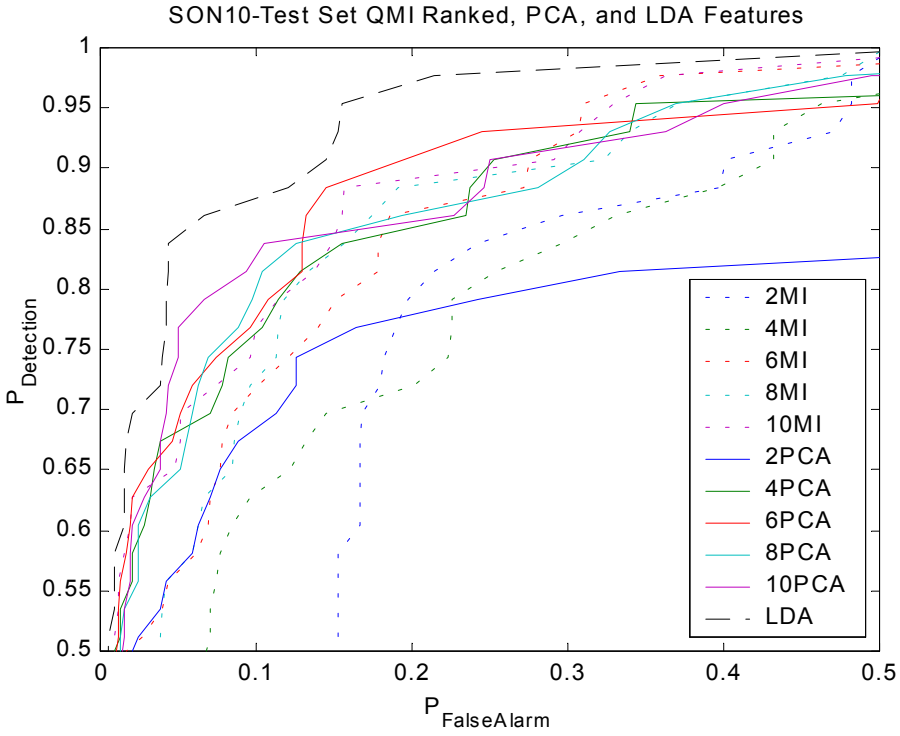

Figure 8-11. Classification performance of QMI ranked features with backward elimination for SON10 test set.

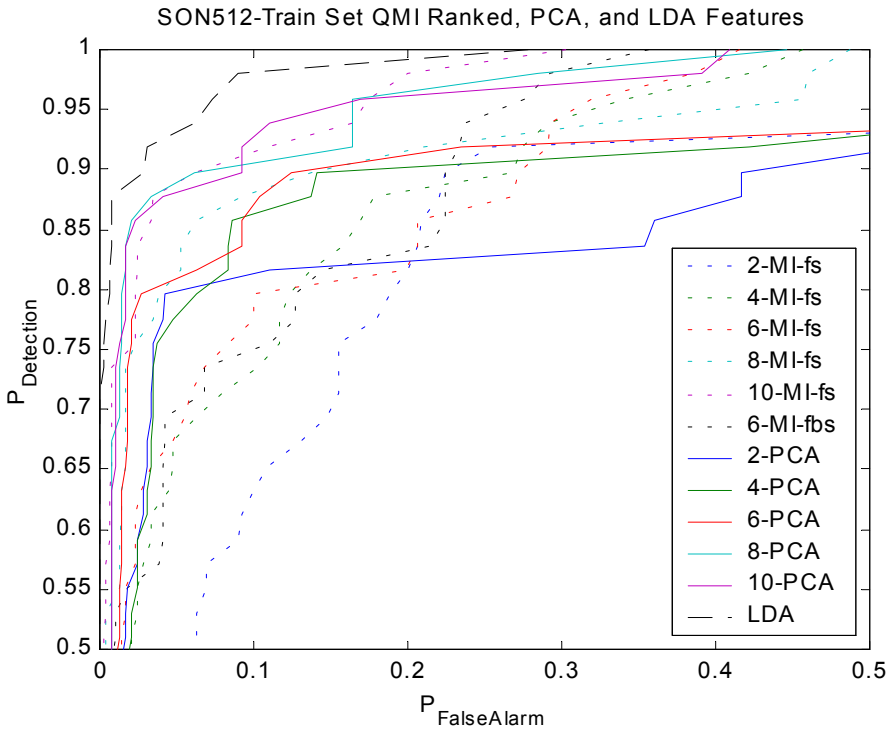

Figure 8-12. Classification performance of QMI ranked features with forward selection for SON512 training set.

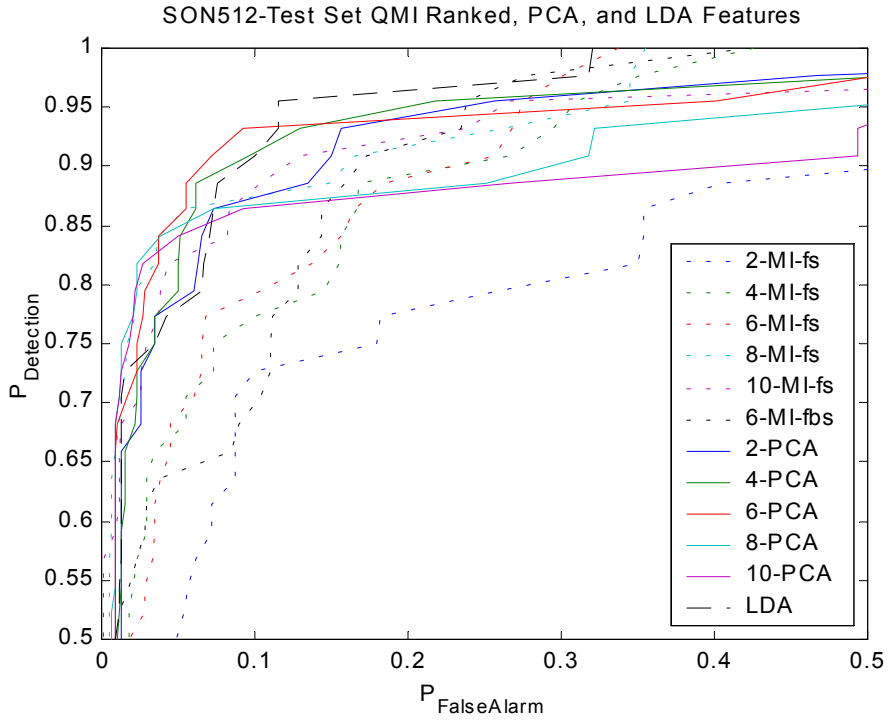

Figure 8-13. Classification performance of QMI ranked features with forward selection for SON512 test set.

Discriminative Feature Mapping

In this section, feature mapping is applied in a discriminative manner. Discriminative feature extraction, or feature extraction for classification, takes into account class-membership information to guide the extraction of new features in an attempt to emphasize or preserve discriminatory information. The most common approach used for this type of feature extraction is LDA. This section presents an approach to discriminative feature mapping for sonar ATR based on maximization of mutual information between class labels and the extracted feature space.

Approach

The goal of this approach is to extract features that emphasize the discriminatory information between the data classes using the ClassInfoFilt criterion. The objective is to extract the minimum dimensional set of novel features that still provides for the desired performance in terms of class separability between targets and false alarms. By maximizing the mutual information between the newly extracted features and the class labels, it is expected that the output feature space will cluster members of the same class together so that subsequent processing (i.e., the classifier stage) will be able to partition the classes more easily. In this approach, both linear (LQMIC) and nonlinear (NLQMIC) mappings are explored. Both methods take all 60 baseline features as input into either a 60-to-M_0 linear combiner or 60-X-M_0 MLP, where X varies from 2 to 6. Both are trained with the CS-QMI between the class labels and the feature space. The output is an M_0-dimensional space of features suited for class discrimination.

In order to improve the training efficiency of the method, the batch training approach from Chapter 6 is used to divide the training set into smaller batches of 200 samples using a special batch selection method. Since there are a disproportionately

larger number of false alarms than targets in these sonar datasets, the batch selection method uses the entire target set all the time with the remainder of the 200 samples randomly selected from among the false alarms. The number of batches (NB) is 50 and the number of epochs per batch (NEB) is three. Both the ITL LM and the SCG advanced training algorithms from Chapter 5 were used to search the parameter space with a fixed kernel size of 0.1. The entire process is shown in Figure 8-14.

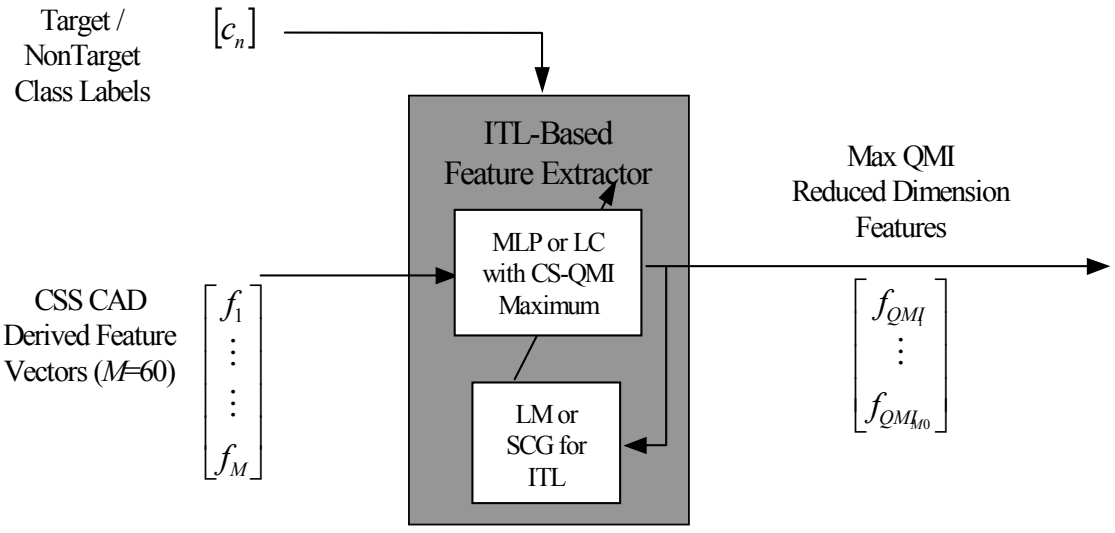

Figure 8-14. Maximum QMI discriminative feature extraction.

Results

In order to compare the discriminative performance of the extracted feature sets, a MVG classifier is used to generate ROC curves for each of the cases. Although the MVG classifier might not be the ideal classifier for QMIC feature extraction, it was chosen in this case and the previous cases because of its simplicity and deterministic nature. ROC curves are also generated for PCA features of various dimensions and for the features generated by the LDA method. Since the parameter search process is

stochastic in nature, ten runs for each feature extraction topology were run. The run with the highest mutual information was selected for each topology.

Figures 8-15 through 8-18 show the detection performance of the QMIC extracted feature sets (dotted lines) of various dimensions compared to the PCA (solid lines) and LDA techniques (dashed lines). Figures 8-15 and 8-16 show the curves for the SON10 training and test sets, respectively, using the linear network with the QMI criterion (LQMIC) for M_0=1,2,3,4. First, notice that in all cases of both training and test sets, the LQMIC outperforms the PCA features of the same dimension as might reasonably be expected. In the training set, all the LQMIC features are also outperform LDA for high values of $P_{Detection}$. In the test set, however, only in the case where M_0=1 does the LQMIC generalize better than LDA. The nonlinear results (NLQMIC) for this dataset are not shown; however, their performance generally follows the same trends as the linear case with greater variability among the 10 runs. Hence, the best NLQMIC runs have comparable to slightly better performance than the LQMIC with the same dimension, while the worst NLQMIC runs performed substantially worse than the same dimensional LQMIC case.

Figures 8-17 and 8-18 show the curves for the SON512 training and test sets, respectively, using LQMIC. Again, for this dataset, the LQMIC features outperform their corresponding PCA features in all cases. In contrast to the prior case, the LDA method outperforms the LQMIC in the training set while the converse is true in the test set. For the SON512 dataset, the NLQMIC criterion again performed similarly in the best case, but with greater variation than the LQMIC. In general, for both of these datasets, increasing the dimensionality of the QMI feature sets provides little improvement overall.

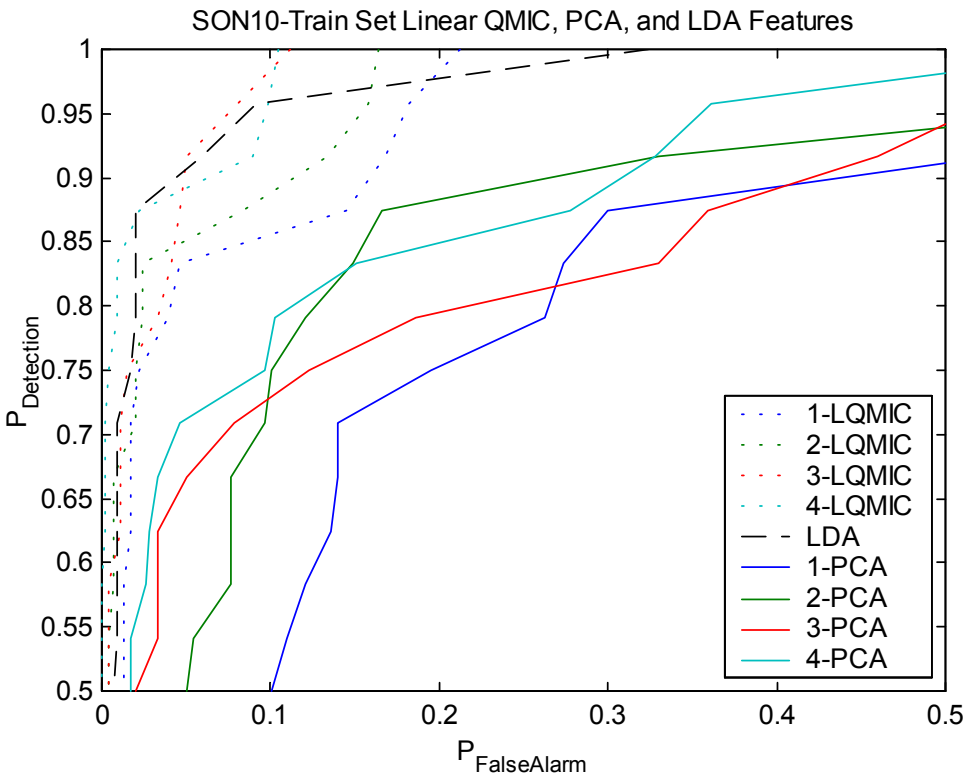

Figure 8-15. Maximum Linear QMI feature extraction for SON10 training set.

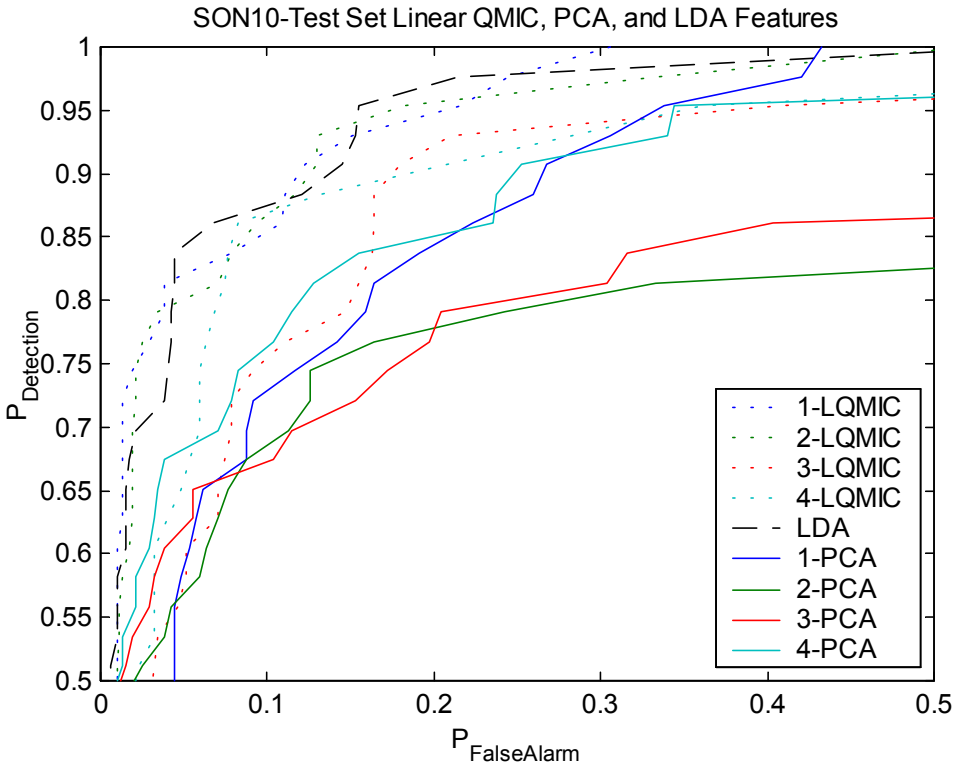

Figure 8-16. Maximum Linear QMI feature extraction for SON10 test set.

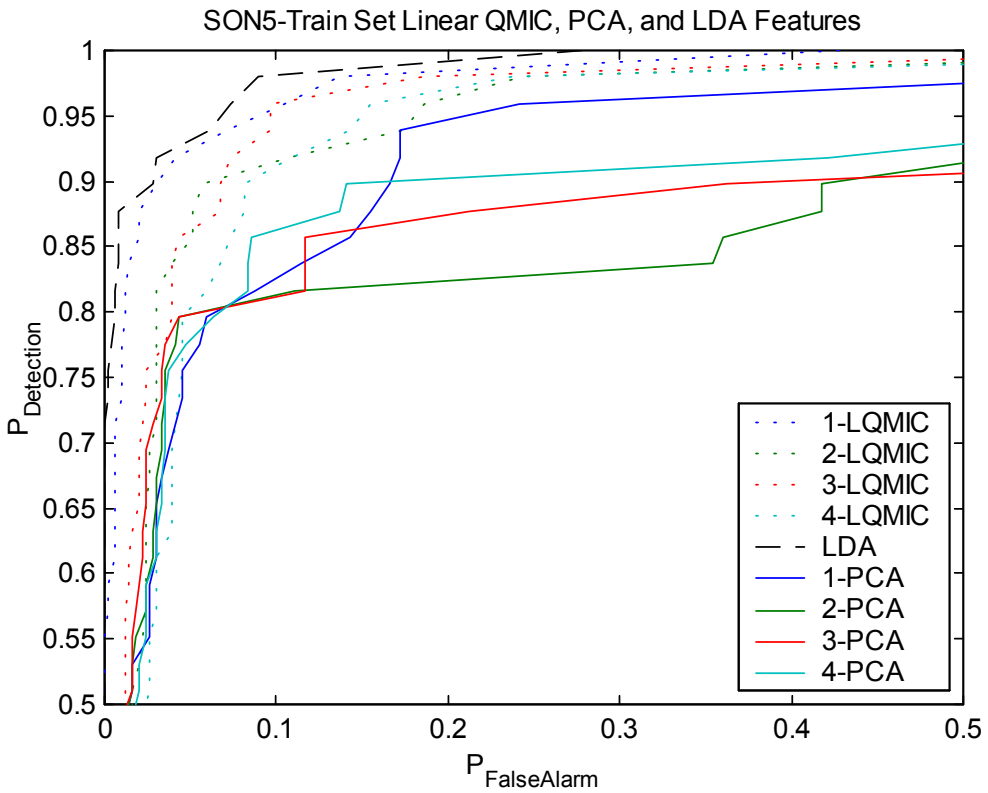

Figure 8-17. Maximum Linear QMI feature extraction for SON512 training set.

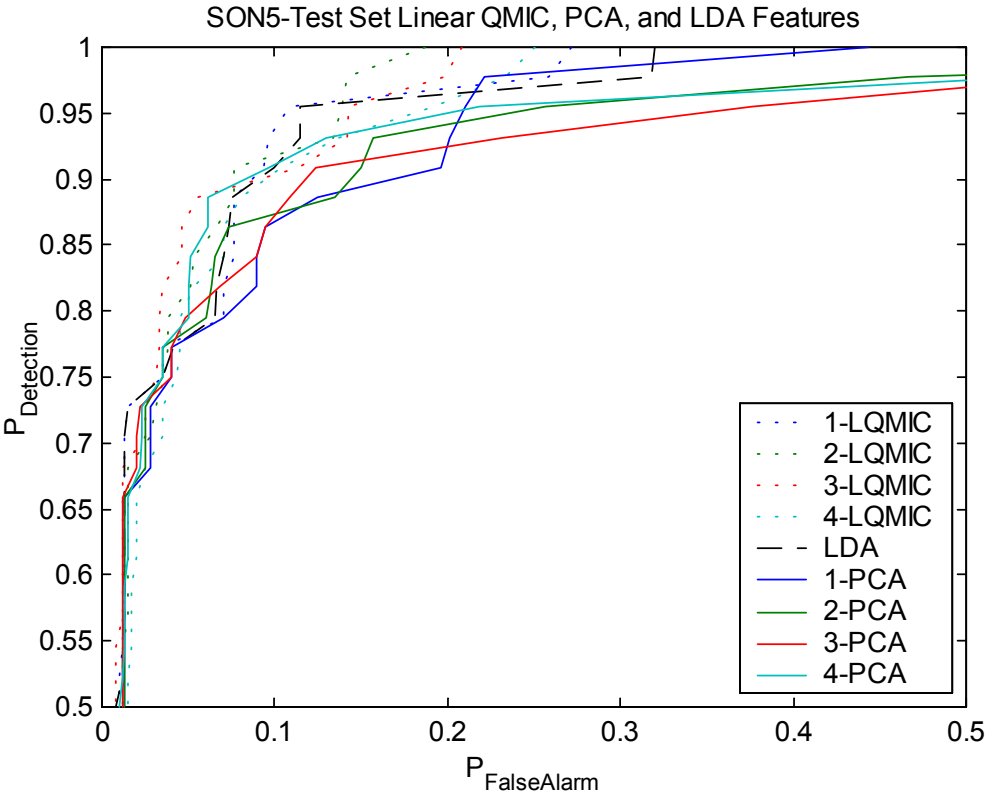

Figure 8-18. Maximum Linear QMI feature extraction for SON512 test set.

Although the added dimensions improve the training performance to a limited extent, the sparseness of available training data make the generalization performance at higher dimensions significantly poorer. For this reason, subsequent examples will be constrained to the cases where M_0=1 or 2.

Figures 8-19 and 8-20 show one example of the NLQMIC feature extractor on the SON10 dataset where M_0=2. Figure 8-19 shows the distribution of the output of the MLP prior to training. The target classes are in shades of red and the false alarms are in shades of blue. Notice the heavy overlap of the classes in the center of the mass. Figure 8-20 shows the distribution of the network output after CS-QMI training. Notice the separation that has transpired with the majority of target samples moving towards the lower boundary, while the false alarms appear more uniformly distributed.

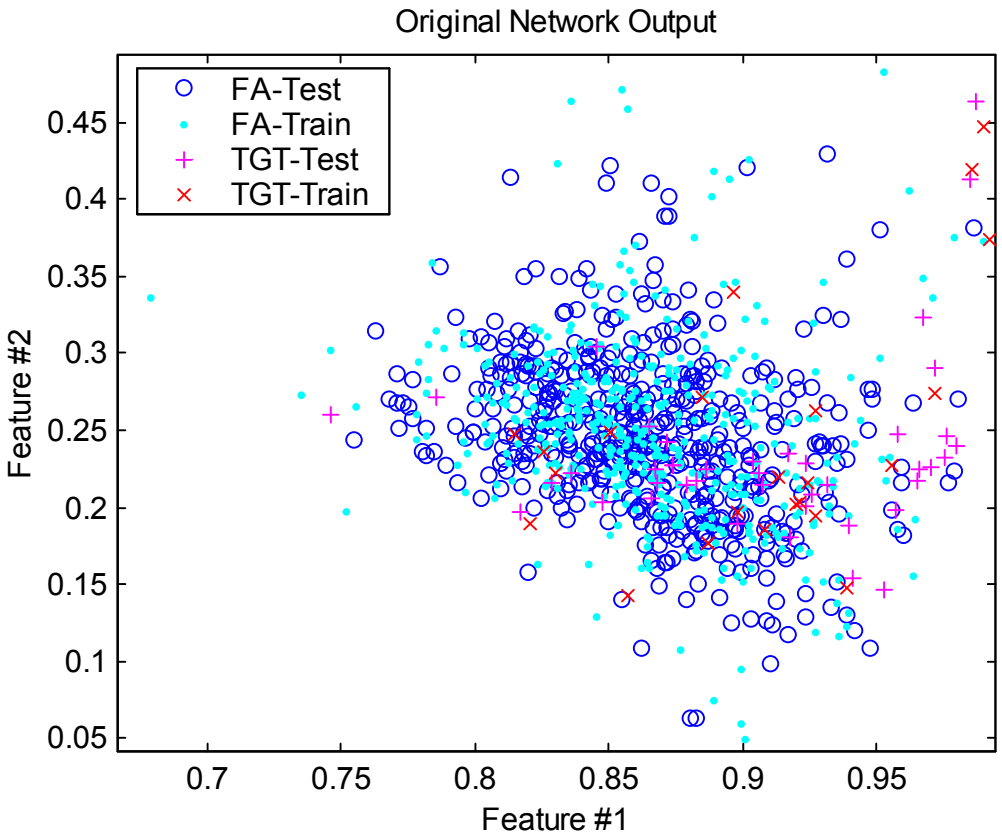

Figure 8-19. Class feature space in 2-D for SON10 prior to training.

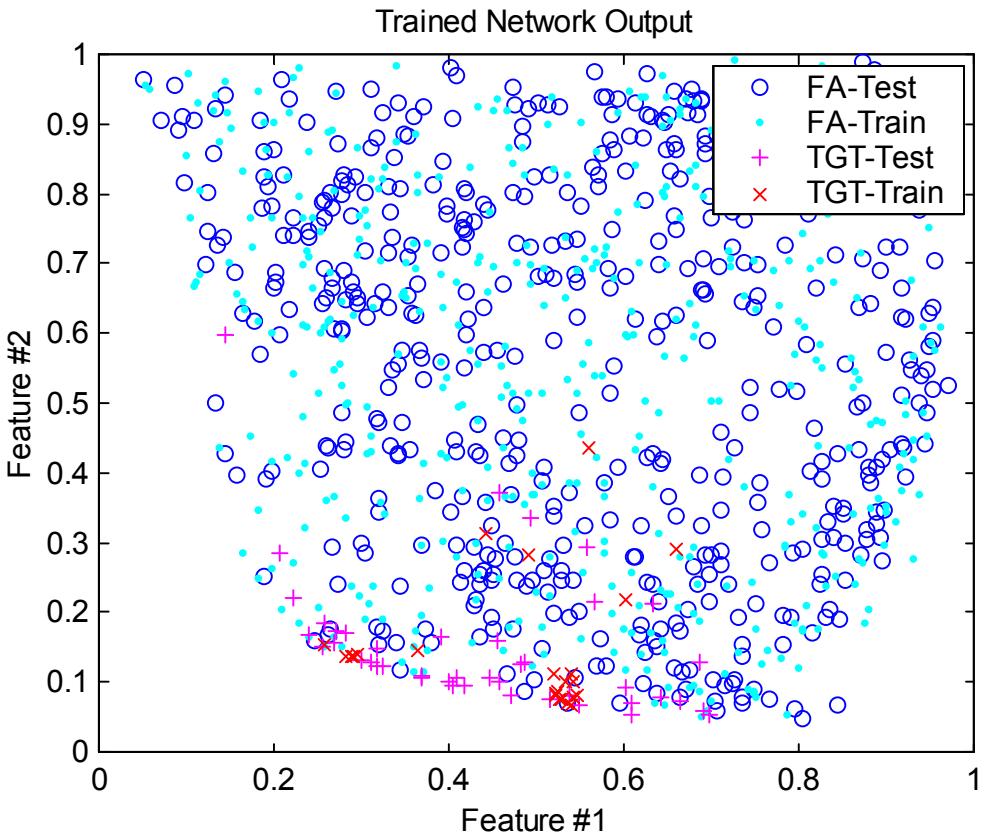

Figure 8-20. Class feature space in 2-D for SON10 after NLQMIC training.

Figures 8-21 and 8-22 show the distributions of the LQMIC and NLQMIC feature spaces for the SON10 and SON512 datasets, respectively, in the case where M_0=1. These figures show the distributions of the both the target and false alarm classes before and after training for both the LQMIC and NLQMIC cases. Note that despite heavy overlap prior to training, both methods perform well at separating the classes. The additional flexibility of the nonlinear method appears to allow better final separation than the linear case. In the NLQMIC case for SON512, the final distribution suggests that a classifier other than MVG might provide better classification performance because of the second mode for the false alarm class that is visible on the right side of the figure. These figures were generated by conducting ten runs using the combined training and test sets for network training. The figures show the individual runs with the highest QMI.

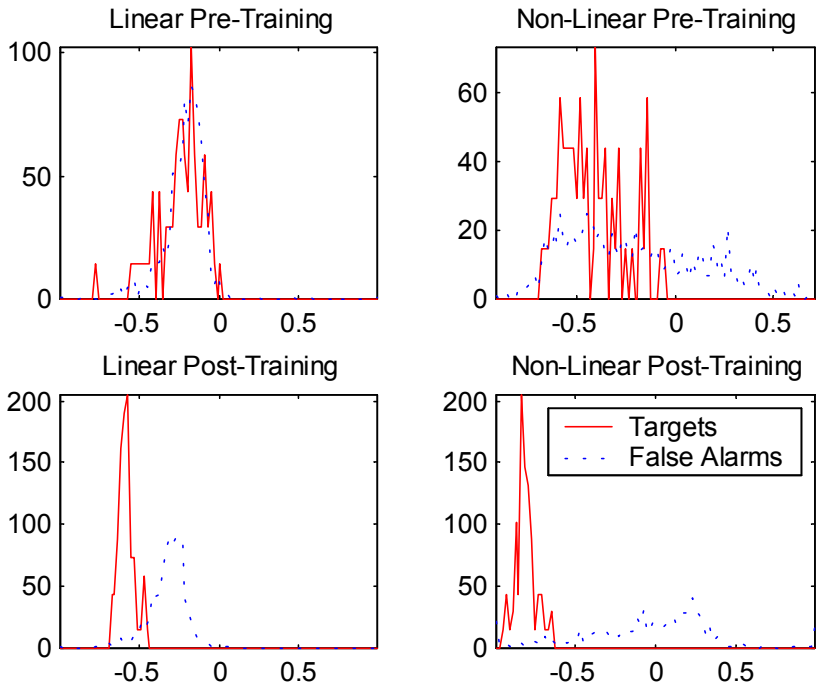

Figure 8-21. Comparison of 1-D PDF for SON10 maximum QMI feature extraction.

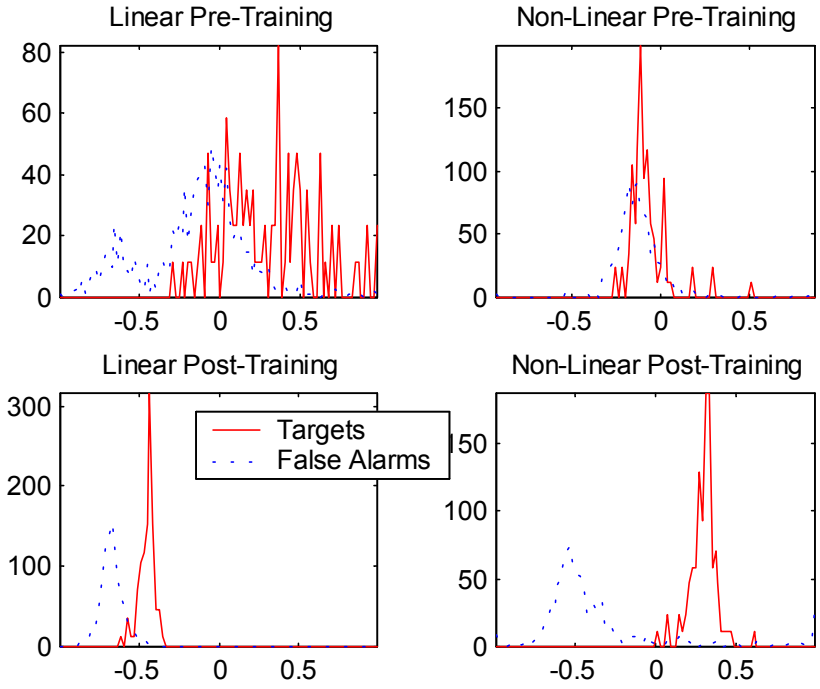

Figure 8-22. Comparison of 1-D PDF for SON512 maximum QMI feature extraction.

Since the SON10 and SON512 datasets are sparsely populated in terms of the number of targets available for training, the above examples show marginal benefit in

applying the LQMIC and NLQMIC methods as compared to a more simplistic approach, such as LDA. In an effort to address this issue using these datasets, the LDA, LQMIC, and NLQMIC methods are applied to extract features using the combined training and test sets for the case where M_0=1 for each of the two datasets. Then, ROC curves are generated for each case. For the LQMIC and NLQMIC methods, the best run of ten is selected based on the QMI calculation. It should be noted that the best ROC performance does not necessarily correspond to the highest QMI. This is the case for most discriminative feature extraction criteria as they are generally used as a proxy for downstream classification performance. The approach of training and testing using the same dataset is also know as the resubstitution method and is often used to provide a lower bound on the classifier error [Fuk90]. The results of this method are shown in Figure 8-23 with the LDA in black, the LQMIC in blue, and the NLQMIC in red for both SON10 and SON512. Notice that both QMIC methods perform better than the LDA with the NLQMIC performing the best overall. This suggests that the QMI criterion, for these datasets and for high values of $P_{Detection}$, is a better criterion than the ratio of between-class scatter and within-class scatter, which is the criterion typically used in LDA.

The final two figures compare the LQMIC and NLQMIC methods with the CSS-Navy ATR algorithms for the SON10 and SON512 datasets. The Navy algorithms essentially designs two classifiers, the KNN and ODFC, based on the 60-dimensional feature space and find the optimum ANDing thresholds as a function of $P_{Detection}$. Typically, the two classifiers are designed by selecting the optimal features using the test set to guide the feature selection process. In this case, it was desired to have a blind test set for comparison purposes, so the classifiers are designed using only the training set.

The LQMIC and NLQMIC feature extractors are the same as described previously for the case where M_0=1 and were trained using the training set exclusively as well. The MVG classifier is used to generate the ROC curves for the QMIC features. The figures show the composite CSS-Navy results in blue and cyan, the individual classifiers for one case in green, the LDA method in black, and the QMIC methods in red. Figure 8-24 shows the case for SON10. Note that although one of the composite classifiers performs the best, the QMIC features perform very well for high $P_{Detection}$. Figure 8-25 shows the SON512 dataset. In this case, NLQMIC features outperform the composite CSS-Navy algorithms for a wide range of $P_{Detection}$, and both QMIC methods perform well in comparison to the individual Navy classifiers.

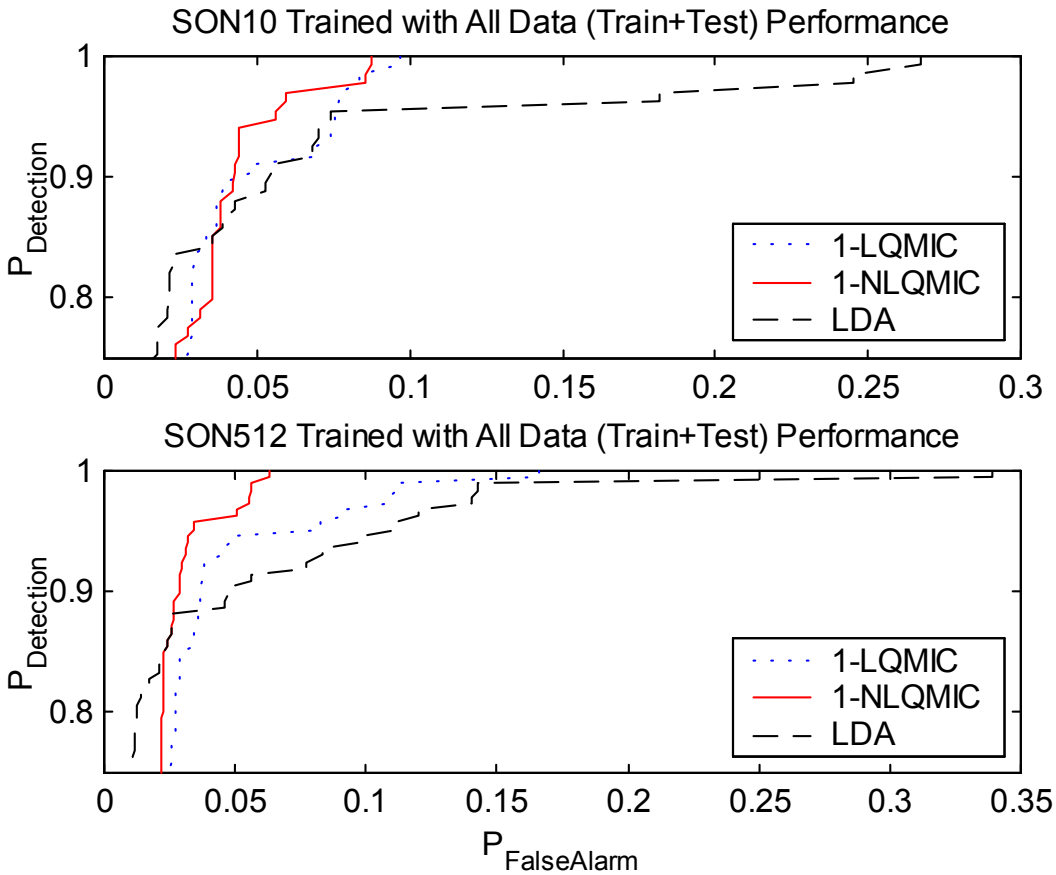

Figure 8-23. Minimum error bound comparison of LDA and QMI feature extraction.

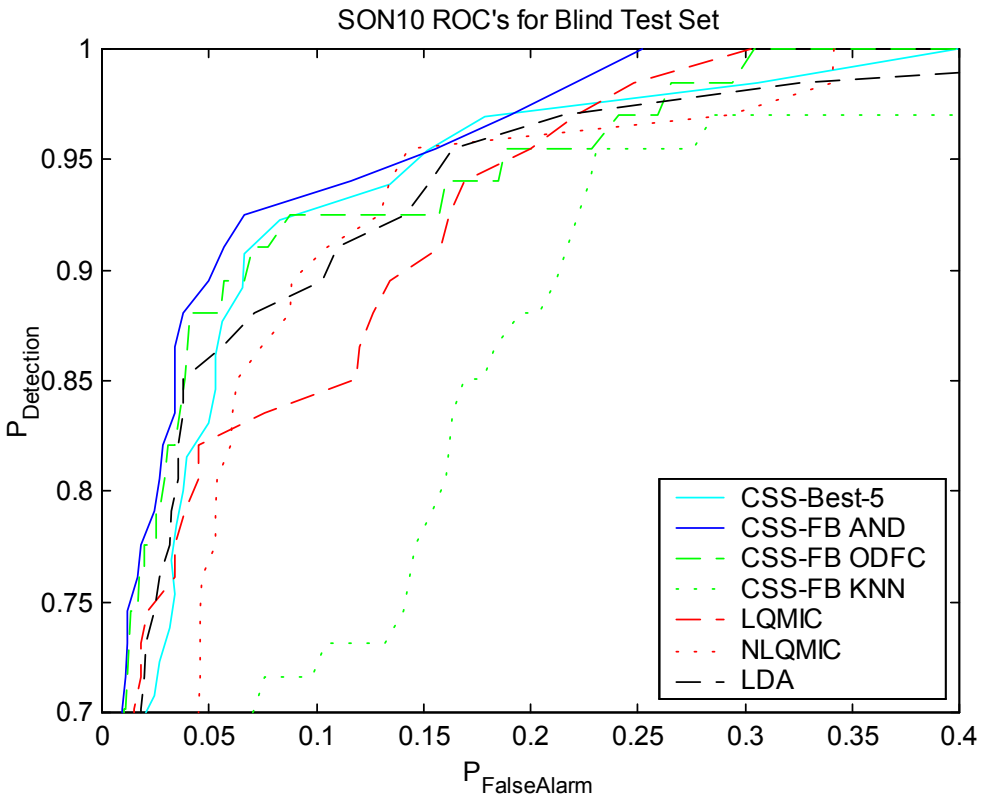

Figure 8-24. Comparison of CSS algorithms and QMI feature extraction for SON10.

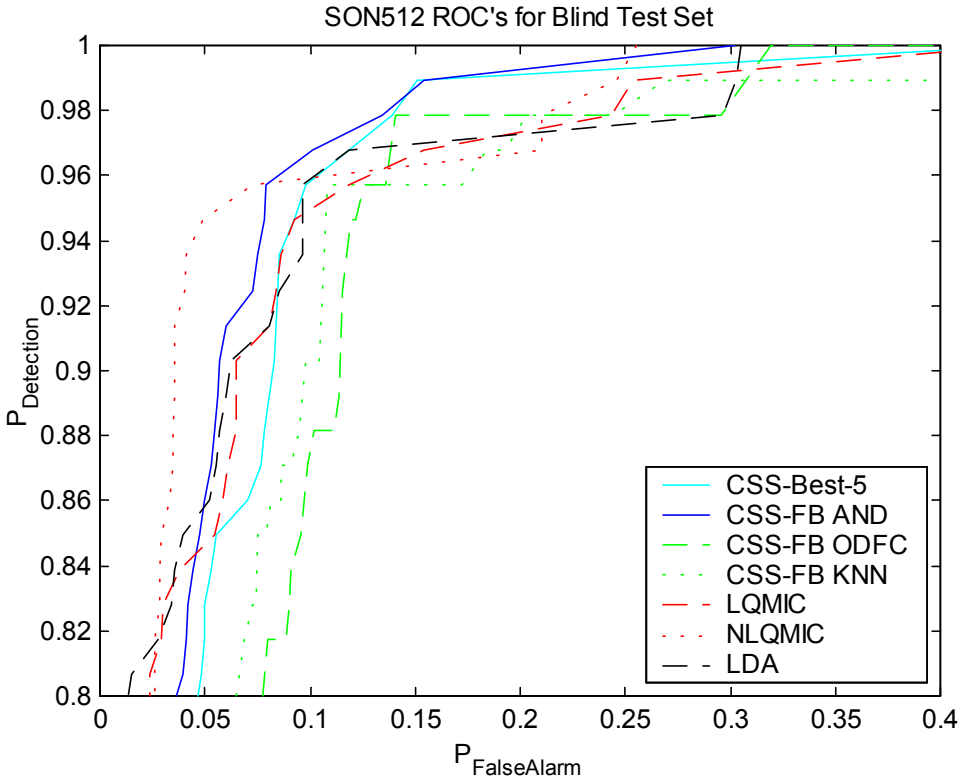

Figure 8-25. Comparison of CSS algorithms and QMI feature extraction for SON512.

Summary of Results

Tables 8-3 and 8-4 summarize the results of the three information-theoretic approaches for feature extraction presented above, along with the results for the PCA, LDA, and CSS methods for the SON10 and SON512 datasets, respectively. The table shows the probability of false alarms, P_{FA}, for each method over a range of detection probabilities, P_D, for the training, test, and complete datasets.

Table 8-3. Summary of detection performance for SON10.

Feature Extraction Method	Total P_{FA}			Training P_{FA}			Test P_{FA}		
	P_D=1.00	P_D=0.95	P_D=0.90	P_D=1.00	P_D=0.95	P_D=0.90	P_D=1.00	P_D=0.95	P_D=0.90
PCA *M*=1	0.57	0.43	0.30	0.58	0.54	0.44	0.43	0.33	0.27
MaxEnt *M*=3	0.57	0.33	0.22	0.47	0.32	0.23	0.51	0.32	0.22
LDA *M*=1	0.56	0.16	0.11	0.32	0.09	0.04	0.56	0.15	0.14
QMI-FBS *M*=5	0.23	0.17	0.14	0.13	0.08	0.07	0.21	0.19	0.16
LQMIC *M*=1	0.30	0.19	0.14	0.24	0.18	0.16	0.31	0.21	0.12
NLQMIC *M*=1	0.34	0.14	0.09	0.06	0.06	0.05	0.33	0.26	0.14
CSSBest5AND	0.40	0.15	0.07	0.03	0.02	0.01	0.39	0.17	0.13
CSSBest5KNN	1.00	0.45	0.30	0.07	0.02	0.02	1.00	0.60	0.42
CSSBest5ODFC	0.51	0.19	0.09	0.10	0.10	0.09	0.50	0.21	0.16
CSS-FB-AND	0.25	0.14	0.05	0.00	0.00	0.00	0.26	0.19	0.10
CSS-FB-KNN *M*=15	0.48	0.23	0.21	0.01	0.01	0.01	0.48	0.28	0.22
CSS-FB-ODFC *M*=23	0.30	0.19	0.07	0.00	0.00	0.00	0.31	0.24	0.16

The first two rows list the unsupervised methods of PCA and MaxEnt for the best overall case of each. It is interesting that the best PCA dimension is the case where *M*=1 for both datasets. This suggests again that more complex classifier designs may be required to obtain better performance for these datasets. However, when compared to the MaxEnt approach to feature extraction, the PCA results are roughly equivalent in some

cases but slightly poorer in most cases. The best MaxEnt dimensionalities are *M*=3 and *M*=5 for the two datasets, respectively.

Table 8-4. Summary of detection performance for SON512.

Feature Extraction Method	Total P_{FA}			Training P_{FA}			Test P_{FA}		
	P_D=1.00	P_D=0.95	P_D=0.90	P_D=1.00	P_D=0.95	P_D=0.90	P_D=1.00	P_D=0.95	P_D=0.90
PCA *M*=1	0.73	0.21	0.18	0.72	0.21	0.17	0.44	0.21	0.17
MaxEnt *M*=5	0.58	0.23	0.14	0.60	0.22	0.13	0.37	0.24	0.16
LDA *M*=1	0.30	0.10	0.06	0.28	0.07	0.03	0.32	0.12	0.09
QMI-FBS *M*=6	0.43	0.25	0.22	0.36	0.26	0.23	0.42	0.24	0.17
LQMIC *M*=1	0.43	0.10	0.07	0.43	0.09	0.03	0.27	0.11	0.09
NLQMIC *M*=1	0.26	0.06	0.04	0.02	0.02	0.02	0.24	0.20	0.08
CSSBest5AND	0.39	0.09	0.07	0.07	0.06	0.05	0.41	0.14	0.10
CSSBest5KNN	0.88	0.16	0.15	0.12	0.12	0.12	0.89	0.31	0.17
CSSBest5ODFC	0.42	0.15	0.11	0.18	0.14	0.10	0.44	0.15	0.12
CSS-FB-AND	0.30	0.08	0.06	0.06	0.04	0.04	0.32	0.13	0.09
CSS-FB-KNN *M*=12	0.90	0.11	0.10	0.09	0.09	0.09	0.94	0.21	0.11
CSS-FB-ODFC *M*=8	0.32	0.12	0.11	0.10	0.10	0.08	0.32	0.16	0.14

The next lines in the table show the LDA method and the best case of the QMI ranked feature selection method. For both datasets, the best case was represented by the combined forward and backward search using the compensation for dimensionality differences. In these cases the dimensionalities were *M*=5 and *M*=6, respectively. For both datasets, these feature sets were among the top performers and for the SON10 data set this approach produced the lowest false alarm rate and best generalization performance for P_D=1.

The next two rows show the both the linear and nonlinear QMIC feature extraction results. For both datasets and both methods, the best dimensionality is *M*=1.

As in the previous discussion, the runs with the highest QMI were selected to produce the result for these tables. Note that both of these are among the top performing methods overall and the NLQMIC produces the best results for the SON512 dataset.

The last six lines in each table summarize two of the CSS approaches. The first, CSSBest5, represents an earlier design of the CSS algorithm that selects the best five features for each of two classifiers, KNN and ODFC, incrementally. The first line shows the combined results while the grayed rows show the intermediate individual classifier results. For these classifiers, each has a dimensionality of M=5. The second method, CSS-FB, replaced the best-five approach with a combined forward and backward search through the feature space. This results in an arbitrary number of features for each classifier ranging from M=8 to M=23 for these datasets. Note that the CSS combined results for both methods are typically among the best in the table. However, the individual classifiers, particularly the KNN, are typically outperformed by many of the other methods in the table. This suggests that some of the information-theoretic methods presented here might be immediately useful in improving the overall detection performance of the CSS methods when used in addition to or in place of the existing classifiers.

Conclusions

The results from this chapter clearly demonstrate the utility and applicability of information-theoretic methods to a complex and challenging real world problem. By applying the fundamental concepts of information-theoretic learning, several methods for addressing the feature extraction problem in the context of sonar ATR have been identified. By leveraging the computational advances of previous chapters, the prognosis for applying these methods to this problem has changed from computationally prohibitive

to completely possible. First, an informative method for feature extraction based on maximum entropy for sparsely sampled datasets was proposed that combines the dimension reduction of PCA with the compressive capacity of a nonlinear information-based mapping function. These informative features provided superior retention of both absolute and discriminative information over a range of several dimensions. Next, an information-theoretic criterion for feature selection was demonstrated along with a novel approach for conducting a combined forward/backward feature selection search. This method produced feature subsets that generally outperformed their PCA counterparts and produce comparable results to using LDA. This method can also provide utility in evaluating the relevance of new candidate feature sets. Lastly, an approach for extracting discriminative features for sonar classification based on mutual information was presented. Although somewhat hampered by the sparseness of available data, these features provide better performance for high probabilities of detection than LDA in most cases and appear to have greater potential for sonar classification than LDA. Additionally, the QMIC features compare favorably with existing Navy feature extraction and classification algorithms providing the potential for increasing overall system performance.

CHAPTER 9
CONCLUSIONS

This chapter provides a summary of the contributions and issues addressed throughout the course of this dissertation. At the inception of this research, the original thoughts were to investigate the applicability of information theory across a wide range of signal processing tasks, as they exist throughout the sonar automatic target recognition problem. These expectations were rapidly recalibrated as the impact of the computational complexity of information-theoretic criteria when combined with large datasets was witnessed firsthand. What resulted was a refocusing of the primary efforts onto the computational aspects of the ITL process with a secondary goal of application to sonar ATR feature extraction.

This research identified three methods that can be used individually or in concert to greatly reduce the computational demands of the ITL process. First, by examining the form of the ITL criteria and applying fundamental algorithm optimization techniques, the complexity of an individual criterion evaluation was reduced up to fifty-fold in terms of computational time. Next, methods to increase the efficiency of the optimization or learning process were examined. By tailoring a suite of the most commonly used parameter search algorithms to account for the nuances of ITL, a collection of advanced training algorithms for ITL were developed. These algorithms demonstrated increases in training computational requirements of up to two orders of magnitude. Finally, in order to ease the quadratic complexity of the native ITL process, a batch training approach was presented that partitions the training data into smaller subsets. This method resulted in

increased training efficiency of several orders of magnitude in some cases. This batch training approach was also shown to be a generalization of the stochastic information gradient method. These three approaches have contributed greatly to the practical utility of ITL and have enabled experimentation that was previously considered prohibitive.

One of the other difficulties that emerged through the course of experimentation was the lack of understanding about causal relationships between the information estimates and the parameters on which the estimates are based. In order to provide some insight as to these relationships, Chapter 7 was dedicated to the exploration of the properties of the information estimators and their parameters. This exploration not only shed light on the relationships between the parameters and the estimates, but also resulted in the development of several novel information-based metrics and methods. First, a scale-invariant form of the entropy metric was derived. Next, a relative entropy measure was introduced allowing for the comparison of entropy estimates derived from different parameter settings. Finally, a method for using the relative entropy measure for the estimation of intrinsic dimensionality was presented with robust results for a wide range of systems.

In Chapter 8, the original goal of applying ITL to sonar ATR was addressed in the context of feature extraction. By applying the concepts of information theory to the feature extraction problem, three information-based methods were identified. These methods were successfully applied to a pair of real-world sonar datasets. First, an informative feature extraction method was demonstrated to have superior performance to the PCA method. Next, the mutual information estimator was shown to be a viable criterion to drive a feature ranking and subset selection scheme. Finally, an approach to

discriminative feature extraction demonstrated a better performance potential than LDA and results comparable to the existing Navy ATR algorithms.

In looking toward the future, several topics warrant continued research and experimentation. The first relates to the control of the kernel size in order to avoid local minima. Although the results presented herein support the online adaptation of the kernel size and further reinforce its utility, additional research is needed to develop a robust methodology for controlling the kernel size. Next, the estimators for mutual information warrant additional investigation. The conditions that cause the unexpected behaviors of the QMI estimators in Chapter 7 need to be better understood so that the estimator parameters can be appropriately set for the problem at hand. Finally, the application of information-theoretic principles to the ATR problem in general remains a largely unexplored field.

APPENDIX
IMPLEMENTATION DETAILS FOR TRAINING ALGORITHMS

This appendix summarizes the changes implemented for two of the parameter search algorithms, GD and LMLS, in order to support the training of ITL systems. The steps in the lists below summarize the major procedures of the algorithms. Plain text represents steps from the basic implementation taken from MATLAB. The bold text represents changes that were made to accommodate ITL criteria.

Gradient Descent for ITL

1. Calculate Network State:
   a. Compute Performance using ITL criterion (instead of MSE).
   b. Compute Internal Network Signals (for backpropagation).
   c. Substitute ITL-based Information Forces for Error terms.
2. Calculate weight gradients.
3. Check for Stopping Criteria:
   a. Maximum Epoch Reached.
   b. Performance Goal Reached.
   c. Maximum Time Reached.
   d. Minimum Gradient Reached.
4. Calculate New Weights: $X_{i+1} = X_i + \mu(\text{grad } X_i)$
5. Increment Epoch.
6. Go to Step 1.

Levenberg-Marquardt with Line-Search for ITL

1. Calculate Initial Network State:
   a. Compute Performance using ITL criterion (instead of SSE).
   b. Compute SSE (Based on Target Vector minus Network output)**-not needed**.
   c. Compute Internal Network Signals (for backpropagation).
   d. Substitute ITL-based Information Forces for Error terms.
   e. Set the Kernel Size to Fixed Mode at current value.
2. Calculate Jacobian and Hessian Approximation using Error terms.
3. Check for Stopping Criteria:
   a. Maximum Epoch Reached.
   b. Performance Goal Reached.

c. Maximum Time Reached.
d. Maximum μ Reached.
e. Minimum Gradient Reached.

4. Compute LM Search Direction: $dX_i = -(J^T J + \mu I)^{-1} J^T E_i$
5. **Compute Step Size (α) using Line Search.**
6. Calculate New Weights: $X_{i+1} = X_i + \alpha dX_i$
7. Calculate New Network State:
   a. Compute Performance using ITL criterion (instead of SSE).
   b. Compute SSE (Based on Target Vector minus Network output)-**not needed**.
   c. Compute Internal Network Signals (for backpropagation).
   d. Substitute ITL-based Information Forces for Error Terms.
8. Compare Performance:
   a. If Performance Improves:
      i. Update Network to New Weights.
      ii. Reset the Kernel Size to Desired Adaptive Mode.
      iii. Recalculate New Performance with Adaptive Kernel Size.
      iv. Set the Kernel Size to Fixed Mode at new value for next iteration.
      v. Decrease μ.
      vi. Increment Epoch.
      vii. Go to Step 2.
   b. If Performance Declines:
      i. Increase μ.
      ii. If $\mu <= \mu_{max}$ Go to Step 4.
      iii. Else go to Step 3.

REFERENCE LIST

[Ari97] T. Aridgides, M. Fernandez, and G. Dobeck, "Adaptive 3-dimensional range-crossrange-frequency filter processing string for sea mine classification in side-scan sonar imagery," in *Proc. SPIE'97*, vol. 3079, Orlando, FL, Apr. 1997, pp. 111-122.

[Ari98] T. Aridgides, M. Fernandez, and G. Dobeck, "Adaptive clutter suppression and fusion processing string for sea mine detection and classification in sonar imagery," in *Proc. SPIE'98*, vol. 3392, Orlando, FL, Apr. 1998, pp. 243-254.

[Ari00] T. Aridgides, M. Fernandez, and G. Dobeck, "Fusion of sea mine detection and classification processing strings for sonar imagery," in *Proc. SPIE'00*, vol. 4038, Orlando, FL, Apr. 2000, pp. 391-401.

[Bat92] R. Battiti, "First and second order methods for learning: Between steepest descent and Newton's method," *Neural Computation*, vol. 4, no. 2, pp.141-166, 1992.

[Bel95a] A. J. Bell and T. J. Sejnowski, "An Information-Maximization Approach to Blind Separation and Blind Deconvolution," *Neural Computation*, vol. 7, no. 6, pp.1129-1159, 1995.

[Bel61] R. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton, NJ: Princeton University Press, 1961.

[Bel95b] M. G. Bello, "Markov random-field-based anomaly screening algorithm," in *Proc. SPIE'95*, vol. 2496, Orlando, FL, Apr. 1995, pp. 466-474.

[Bis95] C. M. Bishop, *Neural Networks for Pattern Recognition*. New York, NY: Oxford University Press, 1995.

[Cam00] F. Camastra and A. Vinciarelli, "Intrinsic Dimension Estimation of data: An approach based on Grassberger-Procaccia Algorithm," *Neural Processing Letters*, vol.14, no. 1, pp. 27-34, Aug. 2001.

[Cha99] V. Chandran and S. Elgar, "Detection of sea mines in sonar imagery using higher-order spectral features," in *Proc. SPIE'99*, vol. 3710, Orlando, FL, Apr. 1999, pp. 578-587.

[Cov91] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York, NY: Wiley, 1991.

[Dec96] G. Deco and D. Obradovic, *An Information-Theoretic Approach to Neural Computing*. New York, NY: Springer, 1996.

[Dem00] H. Demuth and M. Beale, *Neural Network Toolbox User's Guide*. Natick, MA: The MathWorks, Inc., 2000.

[Dob97] G. J. Dobeck, J. C. Hyland, and L. Smedley, "Automated detection and classification of sea mines in sonar imagery," in *Proc. SPIE'97*, vol. 3079, Orlando, FL, Apr. 1997, pp. 90-110.

[Dob99] G. J. Dobeck, "Fusing sonar images for mine detection and classification," in *Proc. SPIE'99*, vol. 3710, Orlando, FL, Apr. 1999, pp. 602-614.

[Dob00] G. J. Dobeck, "Algorithm fusion for the detection and classification of sea mines in the very shallow water region using side-scan sonar imagery," in *Proc. SPIE'00*, vol. 4038, Orlando, FL, Apr. 2000, pp. 348-361.

[Erd00] D. Erdogmus and J. C. Principe, "An Error-Entropy Minimization Algorithm for Supervised Training of Nonlinear Adaptive Systems," *IEEE Trans. on Signal Processing*, vol. 50, no. 7, pp. 1780-1786, Jul. 2002.

[Erd01] D. Erdogmus and J. C. Principe, "An On-Line Adaptation Algorithm for Adaptive System Training with Minimum Error Entropy: Stochastic Information Gradient," in *Proc. 3rd Independent Component Analysis and Blind Signal Separation*, San Diego, CA, Dec. 2001, pp. 7-12.

[Erd02] D. Erdogmus and J. C. Principe, "Generalized Information Potential Criterion for Adaptive System Training," *IEEE Trans. on Neural Networks*, vol. 13, no. 5, pp. 1035-1044, Sep. 2002.

[Fuk90] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. San Diego, CA: Academic Press, 1990.

[Gra83] P. Grassberger and I. Procaccia, "Characterization of strange attractors," *Phys. Rev. Lett.*, vol. 50, no. 5, pp.346-349, 1983.

[Guo98] W. Guo and W. G. Szymczak, "Multiresolution neural networks for mine detection in side scan sonar images," in *Proc. SPIE'98*, vol. 3392, Orlando, FL, Apr. 1998, pp. 297-305.

[Guo99] W. Guo and W. G. Szymczak, "Pseudo multsensor fusion schemes for mine detection in side-scan sonar images," in *Proc. SPIE'99*, vol. 3710, Orlando, FL, Apr. 1999, pp. 638-646.

[Hag96] M. T. Hagan, H. B. Demuth, and M. H. Beale, *Neural Network Design*. Boston, MA: PWS Publishing, 1996.

[Her91] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*. Redwood City, CA: Addison-Wesley, 1991.

[Hua00] J. Huang, C. Ciany, M. Broadman, and S. Doran, "Data fusion of computer-aided detection/computer-aided classification algorithms for classification of mines in very shallow water environments," in *Proc. SPIE'00*, vol. 4038, Orlando, FL, Apr. 2000, pp. 413-421.

[Hyl95] J. C. Hyland and G. J. Dobeck, "Sea mine detection and classification using side-looking sonar," in *Proc. SPIE'95*, vol. 2496, Orlando, FL, Apr. 1995, pp. 442-453.

[Jay57] E. Jaynes, "Information theory and statistical mechanics," *Phys. Rev.*, vol 106, no.4, pp. 620-630, 1957.

[Jes99] K. Jespersen, H. B. D. Sorensen, B. Zerr, B. Stage, and B. Haugsted, "Sonar image enhancements for improved detection of sea mines," in *Proc. SPIE'99*, vol. 3710, Orlando, FL, Apr. 1999, pp. 552-558.

[Kol96] D. Koller and M. Sahami, "Toward Optimal Feature Selection," in *Proc. 13th International Conference on Machine Learning*, Bari, Italy, Jul.1996, pp. 284-292.

[Kro99] G. Kronquist and H. Storm, "Target detection with local discriminant bases and wavelets," in *Proc. SPIE'99*, vol. 3710, Orlando, FL, Apr. 1999, pp. 675-685.

[Kul68] S. Kullback, *Information Theory and Statistics*. New York, NY: Dover, 1968.

[Lan00] H. Lange, "Advanced gray-scale morphological filters for the detection of sea mines in side-scan sonar imagery," in *Proc. SPIE'00*, vol. 4038, Orlando, FL, Apr. 2000, pp. 362-372.

[Lau98] B. Lau and T. Chao, "Aided target recognition processing of MUDSS sonar data," in *Proc. SPIE'98*, vol. 3392, Orlando, FL, Apr. 1998, pp. 234-242.

[Lin89] R. Linsker, "An application of the principle of maximum information preservation to linear systems," in *Advances in Neural Information Processing Systems*. San Francisco, CA: Morgan Kaufmann, 1989, pp. 186-194.

[Mol93] M. F. Moller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, vol. 6, pp. 525-533, 1993.

[Mor97] R. Morejon, "Sonar Automatic Target Recognition: A Computationally Efficient Approach," Research Plan for Doctoral Studies, University of Florida, May 1997.

[Nel96] S. R. Nelson and S. M. Tuovila, "Automated recognition of acoustic image clutter," in *Proc. SPIE'96*, vol. 2765, Orlando, FL, Apr. 1996, pp. 122-129.

[Pow77] M. J. D. Powell, "Restart procedures for the conjugate gradient method," *Math. Programming*, vol. 12, pp. 241-254, 1977.

[Pri00] J. C. Principe, J. W. Fisher III and D. Xu, "Information Theoretic Learning," in *Unsupervised Adaptive Filtering*. New York, NY: Wiley, 2000, ch. 7.

[Ren76] A. Renyi, "On Measures of Entropy and Information," in *Proc. 4th Berkeley Symp. Math. Stat. and Prob.*, vol. 1, pp. 547-561, 1961.

[Rie93] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP Algorithm," in *Proc. of the IEEE Intl. Conf. on Neural Networks*, San Francisco, CA, 1993, pp. 586-591.

[Rie94] M. Riedmiller, "Rprop – Description and Implementation Details," Universitat Karlsruhe, FRG, Tech. Rep., 1994.

[Rub97] Y. D. Rubinstein and T. Hastie, "Discriminative vs Informative Learning," in *Proc. Third International Conference on Knowledge Discovery and Data Mining*, Newport Beach, CA, Aug. 1997, pp. 49-53.

[Sac99] A. Sacramone and M. Desai, "Real-time detection of undersea mines: a complete screening and acoustic fusion processing system," in *Proc. SPIE'99*, vol. 3710, Orlando, FL, Apr. 1999, pp. 615-625.

[Szy99] W. G. Szymczak and W. Guo, "Mine detection using model-trained multiresolution neural networks and variational methods," in *Proc. SPIE'99*, vol. 3710, Orlando, FL, Apr. 1999, pp. 559-569.

[Tor00] K. Torkkola and W. Campbell, "Mutual Information in Learning Feature Transformations," in *Proc. 17th International Conference on Machine Learning*, Stanford, CA, Jul. 2000, pp. 1015-1022.

[Xu98] D. Xu, J.C. Principe, J. Fisher and H. Wu, "A Novel Measure for Independent Component Analysis (ICA)," in *Proc. ICASSP'98*, Seattle, WA, May 1998, pp. 1161-1164.

BIOGRAPHICAL SKETCH

Rodney Alberto Morejon was born on June 7, 1966 in Los Angeles, California as the second son of his Cuban immigrant parents, Juan and Teresita. He graduated second in his class from Melbourne High School, Melbourne, Florida in 1984. After 4 years of study at the University of Miami, he graduated Magna Cum Laude with a Bachelor of Science degree in electrical engineering. In 1990, Rodney graduated with a Master of Science degree in electrical engineering from Florida State University, in Tallahassee, Florida. Since 1990, Rodney has been employed by the United States Department of the Navy as an Electronics Engineer at the Coastal Systems Station in Panama City, Florida. In 1993, Rodney was awarded a United States Government Academic Fellowship to begin studies toward a doctoral degree at the University of Florida in Gainesville, Florida. After one year on campus as a full-time student, Rodney has been working remotely on his research as a part-time student from his home and office in Panama City Beach, Florida.