

José C. Principe

# Information Theoretic Learning

Renyi's Entropy and Kernel Perspectives

# **Information Science and Statistics**

*Series Editors:*

M. Jordan  
R. Nowak  
B. Schölkopf

For other titles published in this series, go to  
<http://www.springer.com/series/3816>

José C. Principe

# Information Theoretic Learning

Renyi's Entropy and Kernel Perspectives

José C. Principe  
University of Florida  
Dept. Electrical Engineering  
& Biomedical Engineering  
Gainesville FL 32611  
NEB 451, Bldg. 33  
USA

*Series Editors*

Michael Jordan  
Division of Computer Science  
and Department of Statistics  
University of California, Berkeley  
Berkeley, CA 94720  
USA

Bernhard Schölkopf  
Max Planck Institute  
for Biological Cybernetics  
Spemannstrasse 38  
72076 Tübingen  
Germany

Robert Nowak  
Department of Electrical  
and Computer Engineering  
University of Wisconsin-Madison  
3627 Engineering Hall  
1415 Engineering Drive  
Madison, WI 53706

ISSN 1613-9011  
ISBN 978-1-4419-1569-6 e-ISBN 978-1-4419-1570-2  
DOI 10.1007/978-1-4419-1570-2  
Springer New York Dordrecht Heidelberg London

Library of Congress Control Number: 2010924811

© Springer Science+Business Media, LLC 2010

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

Printed on acid-free paper

Springer is part of Springer Science+Business Media ([www.springer.com](http://www.springer.com))

To the women in my life  
Natercia  
Joana  
Leonor  
Mariana  
Isabel

---

## Preface

This book is an outgrowth of ten years of research at the University of Florida Computational NeuroEngineering Laboratory (CNEL) in the general area of statistical signal processing and machine learning. One of the goals of writing the book is exactly to bridge the two fields that share so many common problems and techniques but are not yet effectively collaborating.

Unlike other books that cover the state of the art in a given field, this book cuts across engineering (signal processing) and statistics (machine learning) with a common theme: learning seen from the point of view of information theory with an emphasis on Renyi's definition of information. The basic approach is to utilize the information theory descriptors of entropy and divergence as nonparametric cost functions for the design of adaptive systems in unsupervised or supervised training modes. Hence the title: *Information-Theoretic Learning* (ITL). In the course of these studies, we discovered that the main idea enabling a synergistic view as well as algorithmic implementations, does not involve the conventional central moments of the data (mean and covariance). Rather, the core concept is the  $\alpha$ -norm of the PDF, in particular its expected value ( $\alpha = 2$ ), which we call the information potential. This operator and related nonparametric estimators link information theory, optimization of adaptive systems, and reproducing kernel Hilbert spaces in a simple and unconventional way.

Due to the pervasive nature of learning, the reading of the material requires prior basic knowledge on a broad set of subjects such as information theory, density estimation, adaptive filtering, pattern recognition, reproducing kernel Hilbert spaces (RKHS), and kernel machines. Because there are few researchers with such broad interests, the first chapter provides, in simple terms, the minimal foundations of information theory, adaptive filtering, and RKHS, while the appendix reviews density estimation. Once the reader is able to grasp these fundamentals, the book develops a nonparametric framework that is rich in understanding, setting the stage for the evolution of a new generation of algorithms of varying complexity. This book is therefore

useful for professionals who are interested in improving the performance of traditional algorithms as well as researchers who are interested in exploring new approaches to machine learning.

This thematic view of a broad research area is a double-sided sword. By using the same approach to treat many different problems it provides a unique and unifying perspective. On the other hand, it leaves out many competing alternatives and it complicates the evaluation of solutions. For this reason, we present many examples to illustrate and compare performance with conventional alternatives in the context of practical problems. To be more specific, the reader will find:

- Information-theoretic cost functions for linear and nonlinear adaptive filtering that have low complexity but are robust to impulsive noise, and extract valuable structure from the error signal
- Information-theoretic cost functions for classification and unsupervised learning and a new principle of self-organization
- A RKHS for ITL defined on a space of probability density functions that simplify statistical inference
- A new similarity function called correntropy that extends the conventional correlation

The book is organized as follows.

Chapter 1 covers the foundations of information theory, an overview of adaptive systems, and also the basic definitions of RKHS.

Chapter 2 presents the foundations of Renyi's entropy, divergence, mutual information, and their estimators based on the information potential. This is a foundational chapter, and readers should spend time understanding the concepts, and practicing with the algorithms for estimating the ITL descriptors directly from data. The chapter concludes with fast computational algorithms.

Chapter 3 develops the idea of error entropy criterion (EEC) minimization and its minimum error entropy (MEE) algorithm to adapt learning systems. An analysis of the cost function is undertaken and key properties of the error entropy criterion are presented. One of the main reasons why the EEC is useful in practical applications is its robustness to outliers. We establish the link between the EEC and Huber's robust statistics through a weighted least squares point of view. In so doing we define a new function called correntropy that can also be used to train adaptive filters and is easier to compute than EEC. Correntropy defines a metric in the data space and it is directly related to entropy. The chapter ends with a method to adapt the kernel size parameter in adaptive systems training.

Chapter 4 develops a set of algorithms to adapt linear filters using MEE. Basically all the practical gradient-based algorithms are covered: the MEE batch algorithm, the MEE recursive information potential that saves computation, the MEE stochastic information gradient (SIG) that mimics Widrow's LMS algorithm, the MEE self adjusting stepsize, and the normalized MEE. We also present a fixed-point algorithm (no stepsize) with higher complexity

but that is much faster because it explores second-order information content of the cost function. The chapter ends with a comparison with the error correntropy criterion, which has practical computational advantages.

Chapter 5 addresses filtering (regression) problems extending the training algorithms for nonlinear systems. We show how to integrate backpropagation with the error entropy costs, so the reader is able by the end of this chapter to train nonlinear systems with entropic costs. Incidentally, this is really the type of systems that benefit from the error entropy cost because most of the time the errors created are non-Gaussian. Comparisons with traditional mean square error cost are provided. A brief overview of advanced search methods with ITL algorithms is also presented.

Chapter 6 changes the focus to classification problems. The techniques necessary to train classifiers with MEE have already been established in Chapter 5, so this chapter addresses the usefulness of error entropy costs for classification, which is a harder problem than regression. Alternatively, non-parametric classifiers using a MAP approach can be easily implemented and work reasonably well in small-dimensional spaces. For classification, the idea of utilizing the dissimilarity between class labels and system output separately (instead of creating the error) is appealing because of Fano's bound. We extend the cost function to include the ITL divergence measures and quadratic mutual information, and show that this alternative cost function is beneficial not only to train classifiers but also for feature selection. The chapter ends with a proof that the classification error can be lower and upper bounded (i.e., can be bracketed) by Renyi's entropy for alpha greater and smaller than one, respectively.

Chapter 7 treats clustering (the simplest of unsupervised learning methods) using ITL divergence measures. First, we discuss the Cauchy-Schwarz divergence measure as a cost function for clustering, bringing out the nice feature that optimal clusters are not necessarily spherical. Then, a gradient descent algorithm is proposed to find the data partition that minimizes this clustering cost function, and its connection to spectral clustering and optimal graph cuts is established. Gaussian mean shift is also framed as the optimization of an ITL cost function. The chapter ends with a novel information cut algorithm for graph clustering.

Chapter 8 reviews several self-organizing principles based on information-theoretic concepts to show the importance of IT descriptors as cost functions for the optimal design of unsupervised learning systems. Then, a new self-organizing principle called the principle of relevant information is presented that yields as special cases, clustering, principal curves, and vector quantization. Finally, the ITL descriptors are utilized to implement the most common forms of self-organizing principles without assumptions about the data PDFs.

Chapter 9 defines a new reproducing kernel Hilbert space on the space of PDFs with an inner product defined by the cross information potential of ITL. This RKHS provides a functional analysis perspective of ITL and

helps us understand links between statistical inference and the RKHS defined for ITL. Moreover, we show the relationship between ITL descriptors and statistical operators used in machine learning in the RKHS defined by the kernel, including an interpretation of support vector machines.

Chapter 10 defines in the space of random variables a novel generalized correlation function named correntropy. We present many properties of correntropy to make clear its statistical meaning. Based on correntropy, we propose the correntropy coefficient that is bounded by unity and zero for independent random variables, unlike the conventional correlation coefficient. By defining the concept of parametric correntropy, we propose a new correntropy dependence measure that obeys most of Renyi's postulates for dependence. We illustrate the use of correntropy in statistical inference problems, such as matched filtering, tests of nonlinear coupling and as a dependent measure between random variables.

Chapter 11 extends the concept of correntropy to random processes. The name can be properly explained in this context because correntropy (built from correlation plus entropy) looks like correlation but the sum over the lags (or dimensions) is the information potential (the argument of the log of Renyi's entropy). We show that the autocorrentropy function is a positive definite kernel and, as such, defines a novel RKHS with interesting properties. It is possible to define a correntropy spectral density that provides a spectral representation that includes, for the first time, second- and higher-order moments of the random process. We end the chapter with a case study to exemplify how to transform optimal linear algorithms to the correntropy RKHS, and a few examples in speech processing, time series analysis, a correntropy Karhunen-Loeve transform and, object recognition.

The appendix completes the book with a review of kernel density estimation and Renyi's entropy estimation.

The author is conscious that such a vast coverage of topics imposes some compromises of breadth versus depth. To help readers with different backgrounds, profiles, and goals the following flowcharts help establish a road map for the book.

Adaptive Systems (including neural networks) Theme

Ch 1 → Ch 2 → Ch 3 → Ch 4 → Ch 5 → Ch 6 → Ch 7 → Ch 8

Unsupervised Learning Theme

Ch 1 → Ch 2 → Ch 7 → Ch 8

RKHS Theme

Ch 1 → Ch 2 → Ch 9 → Ch 10 → Ch 11

Statistical Signal Processing Theme

Ch 1 → Ch 2 → Ch 3 → Ch 9 → Ch 9 → Ch 11

Pattern Recognition Theme

Ch 1 → Ch 2 → Ch 5 → Ch 6 → Ch 7 → Ch 9 → Ch 10

The book is based on a large collection of journal papers and conference proceedings produced by an extraordinary group of PhD students and

CNEL visitors who were smart enough, knowledgeable enough, and brave enough to think outside the box and ask very pertinent questions about the principles ordinarily used in statistical signal processing and machine learning. Their names are: Deniz Erdogmus, Weifeng Liu, Dongxin Xu, Robert Jenssen, Jianwu Xu, Ignacio Santamaria, Kenneth Hild II, Jeongju Han, Kyu-Hwa Jeong, Sudhir Rao, Puskal Pokharel, Rodney Morejon, Antonio Paiva, Sohan Seth, Il Park, and Abhishek Singh.

To demonstrate my appreciation for their work I consider them my coauthors and list their names in the chapters where their main contributions are centered.

The author is grateful to the National Science Foundation, in particular the Electrical, Communications and Cyber Systems Division in the Engineering Directorate which has funded the great majority of this work and the above mentioned students.

Gainesville, Florida

August, 2009.

---

# Contents

<b>1</b>	<b>Information Theory, Machine Learning, and Reproducing Kernel Hilbert Spaces</b>	1
1.1	Introduction	1
1.2	Information Theory	7
1.3	Entropy	10
1.4	Mutual Information	14
1.5	Relative Entropy and Kullback–Leibler Divergence	16
1.6	Information Theory beyond Communications	19
1.7	Adaptive Model Building	24
1.8	Information-Theoretic Learning	29
1.9	ITL as a Unifying Learning Paradigm	30
1.10	Reproducing Kernel Hilbert Spaces	34
1.11	RKHS and ITL	38
1.12	Conclusions	44
<b>2</b>	<b>Renyi's Entropy, Divergence and Their Nonparametric Estimators</b>	47
	<i>Chapter Coauthors: Dongxin Xu and Deniz Erdogmuns</i>	
2.1	Introduction	47
2.2	Definition and Interpretation of Renyi's Entropy	48
2.3	Quadratic Renyi's Entropy Estimator	56
2.4	Properties of Renyi's Nonparametric Entropy Estimators	60
2.5	Bias and Variance of the Information Potential Estimator	70
2.6	Physical Interpretation of Renyi's Entropy Kernel Estimators	73
2.7	Extension to $\alpha$ -Information Potential with Arbitrary Kernels	79
2.8	Renyi's Divergence and Mutual Information	80
2.9	Quadratic Divergences and Mutual Information	84
2.10	Information Potentials and Forces in the Joint Space	88
2.11	Fast Computation of IP and CIP	96
2.12	Conclusion	101

<b>3 Adaptive Information Filtering with Error Entropy and Error Correntropy Criteria .....</b>	103
<i>Chapter Coauthors: Deniz Erdogmus and Weifeng Liu</i>	
3.1 Introduction .....	103
3.2 The Error Entropy Criterion (EEC) for Adaptation .....	104
3.3 Understanding the Error Entropy Criterion .....	105
3.4 Minimum Error Entropy Algorithm .....	111
3.5 Analysis of MEE Performance Surface .....	113
3.6 Error Entropy, Correntropy, and M Estimation .....	123
3.7 Correntropy Induced Metric and M-Estimation .....	128
3.8 Normalized Information Potential as a Pseudometric .....	131
3.9 Adaptation of the Kernel Size in Adaptive Filtering .....	135
3.10 Conclusions .....	139
<b>4 Algorithms for Entropy and Correntropy Adaptation with Applications to Linear Systems .....</b>	141
<i>Chapter Coauthors: Deniz Erdogmus, Seungju Han, and Abhishek Singh</i>	
4.1 Introduction .....	141
4.2 Recursive Information Potential for MEE (MEE-RIP) .....	142
4.3 Stochastic Information Gradient for MEE (MEE-SIG) .....	146
4.4 Self-Adjusting Stepsize for MEE (MEE-SAS) .....	152
4.5 Normalized MEE (NMEE) .....	157
4.6 Fixed-Point MEE (MEE-FP) .....	162
4.7 Fast Gauss Transform in MEE Adaptation .....	167
4.8 Incomplete Cholesky Decomposition for MEE .....	170
4.9 Linear Filter Adaptation with MSE, MEE and MCC .....	171
4.10 Conclusion .....	177
<b>5 Nonlinear Adaptive Filtering with MEE, MCC, and Applications .....</b>	181
<i>Chapter Coauthors: Deniz Erdogmus, Rodney Morejon, and Weifeng Liu</i>	
5.1 Introduction .....	181
5.2 Backpropagation of Information Forces in MLP Training .....	183
5.3 Advanced Search Methods for Nonlinear Systems .....	186
5.4 ITL Advanced Search Algorithms .....	190
5.5 Application: Prediction of the Mackey–Glass Chaotic Time Series .....	198
5.6 Application: Nonlinear Channel Equalization .....	204
5.7 Error Correntropy Criterion (ECC) in Regression .....	208
5.8 Adaptive Kernel Size in System Identification and Tracking ..	213
5.9 Conclusions .....	217

<b>6 Classification with EEC, Divergence Measures, and Error Bounds .....</b>	219
<i>Chapter Coauthors: Deniz Erdogmus, Dongxin Xu, and Kenneth Hild II</i>	
6.1 Introduction .....	219
6.2 Brief Review of Classification .....	220
6.3 Error Entropy Criterion in Classification .....	222
6.4 Nonparametric Classifiers .....	228
6.5 Classification with Information Divergences .....	231
6.6 ITL Algorithms for Divergence and Mutual Information .....	233
6.6.1 Case Study: Automatic Target Recognition (ATR) with ITL .....	237
6.7 The Role of ITL Feature Extraction in Classification .....	246
6.8 Error Bounds for Classification .....	253
6.9 Conclusions .....	260
<b>7 Clustering with ITL Principles .....</b>	263
<i>Chapter Coauthors: Robert Jenssen and Sudhir Rao</i>	
7.1 Introduction .....	263
7.2 Information-Theoretic Clustering .....	264
7.3 Differential Clustering Using Renyi's Entropy .....	266
7.4 The Clustering Evaluation Function .....	267
7.5 A Gradient Algorithm for Clustering with Dcs .....	273
7.6 Mean Shift Algorithms and Renyi's Entropy .....	282
7.7 Graph-Theoretic Clustering with ITL .....	286
7.8 Information Cut for Clustering .....	292
7.9 Conclusion .....	297
<b>8 Self-Organizing ITL Principles for Unsupervised Learning ..</b>	299
<i>Chapter Coauthors: Sudhir Rao, Deniz Erdogmus, Dongxin Xu, and Kenneth Hild II</i>	
8.1 Introduction .....	299
8.2 Entropy and Cross-Entropy Optimization .....	301
8.3 The Information Maximization Principle .....	304
8.4 Exploiting Spatial Structure for Self-Organization .....	307
8.5 Principle of Redundancy Reduction .....	308
8.6 Independent Component Analysis (ICA) .....	310
8.7 The Information Bottleneck (IB) Method .....	313
8.8 The Principle of Relevant Information (PRI) .....	315
8.9 Self-Organizing Principles with ITL Estimators .....	329
8.10 Conclusions .....	349

<b>9 A Reproducing Kernel Hilbert Space Framework for ITL</b>	351
<i>Chapter Coauthors: Jianwu Xu, Robert Jenssen, Antonio Paiva, and Il Park</i>	
9.1 Introduction	351
9.2 A RKHS Framework for ITL	353
9.3 ITL Cost Functions in the RKHS Framework	358
9.4 ITL Estimators in RKHS	360
9.5 Connection Between ITL and Kernel Methods via RKHS $H_v$	364
9.6 An ITL Perspective of MAP and SVM Classifiers	368
9.7 Case Study: RKHS for Computation with Spike Train	376
9.8 Conclusion	383
<b>10 Correntropy for Random Variables: Properties and Applications in Statistical Inference</b>	385
<i>Chapter Coauthors: Weifeng Liu, Puskal Pokharel, Jianwu Xu, and Sohan Seth</i>	
10.1 Introduction	385
10.2 Cross-Correntropy: Definitions and Properties	386
10.3 Centered Cross-Correntropy and Correntropy Coefficient	396
10.4 Parametric Cross-Correntropy and Measures of Dependence	399
10.5 Application: Matched Filtering	402
10.6 Application: Nonlinear Coupling Tests	409
10.7 Application: Statistical Dependence Tests	410
10.8 Conclusions	412
<b>11 Correntropy for Random Processes: Properties and Applications in Signal Processing</b>	415
<i>Chapter Coauthors: Puskal Pokharel, Ignacio Santamaria, Jianwu Xu, Kyu-hwa Jeong, and Weifeng Liu</i>	
11.1 Introduction	415
11.2 Autocorrentropy Function: Definition and Properties	417
11.3 Cross-Correntropy Function: Definition and Properties	424
11.4 Optimal Linear Filters in $H_v$	425
11.5 Correntropy MACE (CMACE) Filter in $H_v$	427
11.6 Application: Autocorrentropy Function as a Similarity Measure over Lags	431
11.7 Application: Karhunen-Loeve Transform in $H_v$	438
11.8 Application: Blind Source Separation	442
11.9 Application: CMACE for Automatic Target Recognition	448
11.10 Conclusion	454
<b>A PDF Estimation Methods and Experimental Evaluation of ITL Descriptors</b>	457
<i>Chapter Coauthors: Deniz Erdogan and Rodney Morejon</i>	
A.1 Introduction	457
A.2 Probability Density Function Estimation	457

A.3	Nonparametric Entropy Estimation .....	472
A.4	Estimation of Information–Theoretic Descriptors .....	474
A.5	Convolution Smoothing .....	494
A.6	Conclusions .....	497
<b>Bibliography</b>	.....	499
<b>Index</b>	.....	517

---

## Abbreviations, Symbols and Notation

AIC	Akaike Information Criterion
ATR	Automatic Target Recognition
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BIC	Bayesian Information Criterion
BFGS	Broyden Fletcher Golfarb Shanno Method
BP	Back Propagation Algorithm
BSS	Blind Signal Separation
BW	Bi-square Weighting
CDF	Cumulative Distribution Function
CDM	Correntropy Dependence Measure
CE	Cross Entropy
CEEC	Centered Error Entropy Criterion
CEF	Clustering Evaluation Function
CG	Conjugate Gradient Algorithm
CGB	Power Beagle Method
CGF	Fletcher Reeves Method
CGP	Polak Ribiere Method
CIM	Correntropy Induced Metric
CIP	Cross Information Potential
CMACE	Correntropy MACE
CMF	Correntropy Matched Filter
COCO	Kernel Constrained Covariance
CoKLT	Correntropy Karhunen Loeve transform
CSD	Correntropy Spectral Density
D <sub>CS</sub>	Cauchy-Schwarz divergence
D <sub>ED</sub>	Euclidean Distance
D <sub>KL</sub>	Kullback-Liebler Divergence
D <sub><math>\alpha</math></sub>	Renyi's Divergence
DFT	Discrete Fourier Transform
EEC	Error Entropy Criterion

ECC	Error Correntropy Criterion
EM	Expectation-maximization Algorithm
FGT	Fast Gauss Transform
FFT	Fast Fourier Transform
FIR	Finite Impulse Response Filter
FLOPS	Floating Point Operations Per Second
GBMS	Gaussian Blurring Mean Shift Algorithm
GCF	Generalized Correlation Function
GD	Gradient Descent Algorithm
GIF	Generalized Information Forces
GIP	Generalized Information Potential
GMS	Gaussian Mean Shift Algorithm
I-Max	Information (spatial) maximization Principle
$I_\alpha$	Renyi's Mutual Information
$I_{CS}$	Cauchy-Schwarz Mutual Information
$I_{ED}$	Euclidean Mutual Information
$I_S$	Shannon Mutual Information
IB	Information Bottleneck Method
ICA	Independent Component Analysis
ICD	Incomplete Cholesky Decomposition
IF	Information Forces
InfoMax	Information Maximization Principle
IP	Information Potential
IT	Information Theory
ITL	Information Theoretic Learning
ITVQ	Information Theoretic Vector Quantization
KICA	Kernel Independent Component Analysis
KLT	Karhunen Loeve transform
LAR	Least Angle Regression
LBG	Linde-Buzo and Gray algorithm
LDA	Linear discriminant Analysis
LM	Levenberg-Marquadt algorithm
LMS	Least Mean Square Algorithm
MACE	Minimum Average Correlation Energy
MAP	Maximum a Posteriori
MaxEnt	Maximization of Entropy Principle
MCC	Maximum Correntropy Algorithm
MCE	Minimum Classification Error
MDL	Minimum Description Length
MED	Maximum Entropy Deconvolution
mED	Minimum Entropy Deconvolution
MEE	Minimum Error Entropy
MEE-BP	Minimum Error Entropy Back Propagation
MI	Mutual Information
MISE	Mean Integrated Square Error
ML	Maximum Likelihood

MLP	Multilayer Perceptron
MinXent	Minimization of Cross Entropy Principle
MMD	Maximum Mean Discrepancy
MMSE	Minimum Mean Square Error
MRMI	Minimization of Renyi's Mutual Information
MSE	Mean Square Error
NLMS	Normalized LMS
OSS	One Step Secant Method
PCA	Principal Component Analysis
PDA	Pitch Detection Algorithm
PDF	Probability Density Function
PE	Processing Element
PMF	Probability Mass Function
PRI	Principle of Relevant Information
PSD	Power Spectrum Density
QMI	Quadratic Mutual Information
QMI <sub>CS</sub>	Quadratic Mutual Information Cauchy-Schwarz
QMI <sub>ED</sub>	Quadratic Mutual Information Euclidean Distance
RBF	Radial Basis Function Network
RKHS	Reproducing Kernel Hilbert Space
RIP	Recursive Information Potential
RLS	Recursive Least Squares
ROC	Receiver Operating Characteristics
RP	Resilient Back Propagation Algorithm
SAR	Synthetic Aperture Radar
SCG	Scaled Conjugate Gradient Algorithm
SDR	Signal Distortion Ratio
SIG	Stochastic Information Gradient
SIR	Signal Interference Ratio
SOM	Self-Organizing Map
SVM	Support Vector Machines
TDNN	Time Delay Neural Network
TIMIT	Texas Instrument- MIT database
WSNR	Weighted Signal to Noise Ratio

## Important Symbols

$V(x)$	Information Potential ( $\alpha = 2$ )
$V_\alpha(x)$	$\alpha$ -Information Potential
$V_C(x)$	Cross Information Potential
$V_J(x)$	Joint Information Potential
$V_M(x)$	Marginal Information Potential
$V_N(x)$	Normalized Information Potential
$V(X,Y)$	Cross Information Potential
$F(x)$	Information Force ( $\alpha = 2$ )

$F_\alpha(x)$	$\alpha$ -Information Force
$v(x,y)$	Correntropy
$v(t,s)$	Correntropy Function
$v_{a,b}(x,y)$	Parameteric Correntropy
$u(x,y)$	Centered Correntropy
$u(t,s)$	Centered Correntropy Function
$E(X)$	Expected Value of $x$
$H(X)$	Shannon Entropy
$H_\alpha(X)$	Renyi's Entropy of order $\alpha$
$I(X)$	Information of $x$
$I(X,Y)$	Shannon Mutual Information
$D(p  q)$	Divergence between $p$ and $q$ (Kullback-Leibler or Renyi)
$D(p,q)$	Other Divergences
$p(x)$	Probability Density Function (or Probability Mass Function)
$\kappa(x)$	Kernel Function
$G_\sigma(x)$	Gaussian Kernel with Bandwidth (or size) $\sigma$
$H$	Hilbert Space
$H_\kappa$	RKHS defined by kernel $\kappa$
$\hat{V}$	Estimated (Information Potential)
$f'$	Derivative
$f''$	Double Derivative
$\nabla$	Gradient Operator
$\partial/\partial x$	Partial Derivative
$\delta(t)$	Delta Function
$net(n)$	Linear part of nonlinear PE
$\varphi(x)$	Sigmoid Nonlinearity

## Notation

There are mainly *three* types of variables we need to distinguish among: scalar, vector, and matrix variables.

The following is a list of the notational conventions used in the book:

1. We use *small italic* letters to denote scalar variables.
2. We use *CAPITAL ITALIC* letters to denote scalar constants and random variables
3. We use **small bold** letters for vectors
4. We use **CAPITAL BOLD** letters to denote matrices
5. We use parentheses to denote the time-dependency of any variables (either scalar, vector or matrix).
6. We use the superscript  $T$  to denote transposition.
7. All variables in our presentation are real, unless explicitly stated.
8. All vectors in our presentation are column vectors without exception.
9. We use subscript indices to denote 1) a component of a vector (or a matrix).

# Information Theory, Machine Learning, and Reproducing Kernel Hilbert Spaces

## 1.1 Introduction

The common problem faced by many data processing professionals is how to best extract the information contained in data. In our daily lives and in our professions, we are bombarded by huge amounts of data, but most often data are not our primary interest. Data hides, either in time structure or in spatial redundancy, important clues to answer the information-processing questions we pose. We are using the term information in the colloquial sense, and therefore it may mean different things to different people, which is OK for now. We all realize that the use of computers and the Web accelerated tremendously the accessibility and the amount of data being generated. Therefore the pressure to distill information from data will mount at an increasing pace in the future, and old ways of dealing with this problem will be forced to evolve and adapt to the new reality. To many (including the author) this represents nothing less than a paradigm shift, from hypothesis-based, to evidence-based science and it will affect the core design strategies in many disciplines including learning theory and adaptive systems.

## Modeling Levels

One productive step involved in extracting information from data is data modeling. A model of the data basically summarizes the process of its generation, and allows for the optimal design of data processing systems. Therefore the overall data processing goal can be thought as the macroscopic modeling level. We pursue here methodologies based on probabilistic reasoning because it has played and will continue to play a central role in the endeavor of extracting information from data. Probability theory provides an established framework to work with uncertain or noisy data. In this framework, learning from samples can be thought of as discovering structure in data, or equivalently,

as finding dependencies in the data generation model under the uncertainty produced by noise, incomplete knowledge of the sample space, and unknown parameters or the data structure.

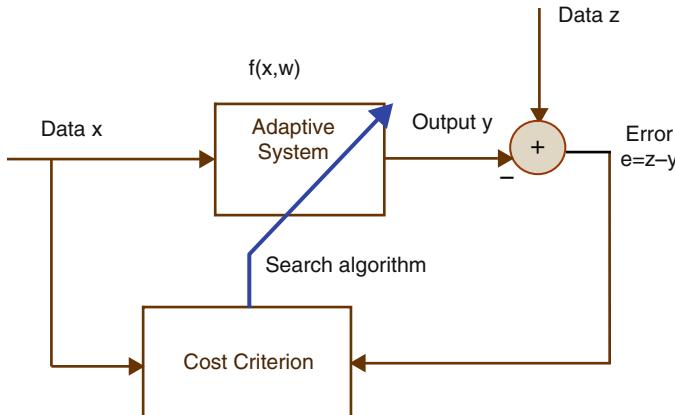
When the data sources contain all the information in their distribution, it is reasonable to build information-processing systems that directly use the data's joint probability density function (PDF) (or any other marginal or conditional PDF, which is easily obtained from the joint distribution). Examples of this approach include the current work on graphical models and its special cases such as the Bayesian filter or the hidden Markov model, which are very powerful but are computationally demanding. Bayesian reasoning allows direct implementation of macroscopic system goals from the data, hence we call this approach microscopic modeling.

Another major line of reasoning in statistics deals with the construction of scalar descriptors of the PDF that, under an appropriate set of modeling assumptions, succinctly characterize the data structure. In our terminology, this is the *mesoscopic modeling level*. The advantage of mesoscopic data descriptors is that they solve the macroscopic data processing goals with computational simplicity, enabling both portable implementations in engineering and “giga-dataset” processing in machine learning. Statistical moments are, in the fields of statistical signal processing and machine learning, by far the most widely used mesoscopic descriptors of the PDF. The appeal of moment expansions is that there are consistent nonparametric estimators for the moments. Moreover, as is well known in probability theory, well-behaved PDFs can be described as accurately as desired by increasing the expansion order. In particular, if the Gaussian assumption is invoked for the PDF, then the mean and the variance completely describe the PDF.

Let us contrast the microscopic and the mesoscopic modeling levels with a simple macroscopic goal of time series modeling. Given a sufficient amount of data, the Bayesian filter is capable of modeling (with a weak Markovian assumption) any time series by directly using the posterior density with a recursive estimator (a state model). Under linearity and Gaussian assumptions, the Kalman filter can provide exactly the same solution based on mesoscopic descriptors (the data covariance function) with the difference that it requires only a fraction of the data and computational resources required to implement the Bayesian filter. Even if the data do not obey these assumptions, the Kalman filter may still be useful because of its computational simplicity and because the Bayesian filter suffers from poor estimation when data are scarce.

This book uses mesoscopic statistical models for adaptive filtering and machine learning. Let us assume that the problem we are facing is to find a relationship between  $\{x, z\}$  by observing pairs of samples  $(x_i, z_i)$  produced by the experimental signal source. Figure 1.1 provides the fundamental building block for this scenario.

A learning machine (linear or nonlinear) having a set of free parameters  $w$  is constructed to receive the data  $x_i$  and produce an output  $y_i$ . We then compare how similar  $y_i$  is to  $z_i$  according to some criterion and minimize this



**Fig. 1.1.** Model building through adaptation.

difference by changing the parameters  $w$  with some systematic procedure. In the end we obtain a system that, when activated with new data from the source  $x$ , approximates the unknown  $z$  with  $y$ . The procedure just outlined builds implicitly a model for the relationship between  $x$  and  $z$ . We call the system a *learning machine* or *adaptive system* and the process of finding parameters from data, *learning* or *adaptation*.

This idea of training adaptive systems is far from new (probably used for the first time by Gauss or Legendre in linear regression), and there are many different approaches to formulate and characterize the training methods. For example, the adaptive system shown in Figure 1.1 uses what is known as supervised learning. Another type of learning, known as unsupervised learning occurs when  $x$  is available, but  $z$  is not. More is said later in the text about supervised and unsupervised learning.

### Parametric and Nonparametric Model Building

Both parametric and nonparametric designs create models from data and the trade-offs between the two have been a central theme in statistical thinking. It is appropriate to recall the debate, summarized in [133], that happened in the early 1920s. In this debate Fisher and Pearson argued about how to model data when neither the error distribution nor the functional form of the fitting function was specified. Fisher proposed to select a simple fitting function (in his terminology the problem of specification) and concentrate on estimating its parameters (in his terminology the problem of estimation). Fisher championed this parametric (or model-based) approach because of its lower variance, in particular when data are scarce. On the other hand, Pearson advocated the importance of specification in detriment of estimation, which leads to nonparametric learning approaches. They handle the bias of the fit well, but have difficulty with the model variance. Looking critically at the

literature in pattern recognition and time series analysis for the last 80 years, Fisher's approach seems the winner because it has been widely utilized in statistics, machine learning, and statistical signal processing and it is the most accurate as long as the assumed model fits the data generation mechanism. Unfortunately, the theory only handles rather simple statistical models that may not be accurate enough to describe the complex PDFs of today's real-world problems.

The learning machine depicted in Figure 1.1 can be designed using either parametric or nonparametric modeling methodologies. The system (also called the mapping function) implements by its topology a set of functions, which defines a manifold of possible solutions depending upon its parametric values. Therefore, the procedure provides in general a parametric model for the statistical relation between input–desired data. Assumptions of linearity (for regression) or Gaussianity (for classification) are imposed so that optimal solutions could be analytically computed, but the set of possible input–output mappings is rather restrictive. Neural networks and kernel machines are both universal mappers; the former combine discriminant functions created by a fixed number of nonlinear units that are incrementally adapted from the data, and the latter functionally map the input data. Therefore, the user does not need to select an explicit parametric model *a priori*, but proper design steps still need to be followed. In the extreme case of placing in each sample a simple fixed kernel and weighting the local contributions, the system effectively implements a nonparametric model. Therefore, the model bias is currently much less important than in the early days of machine learning. Likewise, the benefits of the nonparametric approach are also increasing due to the availability of large amounts of training data (that reduce the variance), added to the ever-increasing power in computational speed.

Finally, the estimation part of the modeling can not be forgotten: (1) the model parameters still need to be learned from the data through an appropriate loss function (criterion); (2) to reach the optimal solution, the learning algorithms need to be able to deal with the nonlinear or the functional nature of the mappings. The block diagram of Figure 1.1 enables online model building, by feeding back the results of the model evaluation to the adaptation algorithm which will seek the extremum of the criterion by continuously modifying the system parameters. This is not the only way of solving the optimization problem, but it is the one that is preferred in this book.

Selecting an appropriate loss function or criterion for the goals of machine learning has been largely neglected, and the parametric/nonparametric modeling issue reappears. Perhaps the most principled approach is to translate the goal of the analysis into a cost function, but this is often complex (i.e., how to minimize the probability of errors in classification). The moments (e.g., the mean square error criterion (MSE) which is the second-order moment of the error PDF) are the most often preferred. The success story of MSE is regression because all the information on the error PDF under a Gaussian assumption is captured in a simple analytical model. However, MSE is used in situations

where it may be a poor descriptor of optimality such as in classification, where classifiers are nonlinear and the errors are not Gaussian distributed. What we should not forget is that the mapping function and the cost function work in tandem, and the overall performance is limited by the one that reflects the poorest approximation for the problem at hand.

## The Information-Theoretical Learning Approach

This book combines a functional mapper with a cost function based on mesoscopic data modeling for computational efficiency. However, the proposed criterion does not suffer from the limitation of Gaussianity inherent in cost functions based on second-order moments (MSE). This is achieved with information-theoretic descriptors of entropy and dissimilarity (divergence and mutual information) combined with nonparametric PDF estimators, which bring robustness and generality to the cost function, and improve performance in many realistic scenarios (fat-tail distributions, and severe outlier noise). One of the other appeals of this information theoretic learning (ITL) methodology is that it can, with minor modifications, use the conventional learning and adaptation methodologies of adaptive filters, neural networks, and kernel learning. Our hope is that this line of research can close the performance gap between microscopic and mesoscopic data modeling, while preserving the simplicity of the latter.

Let us then interpret adaptation in Figure 1.1 from an information-theoretic point of view. The information contained in the joint PDF  $p(x, z)$  should be transferred as efficiently as possible to the parameters  $w$  of the learning system. Therefore, one would hope to extract as much information as possible from the error PDF  $p(e)$  by changing  $w$  to make  $y$  as close as possible to  $z$  in an information sense. Entropy measures uncertainty of the error, therefore the cost should minimize the entropy of the error.

Shannon entropy [293] for continuous random variables reads  $\int p(x) \log p(x) dx$ , therefore knowledge of the data PDF  $p(x)$  is a necessary first step to estimate entropy, divergence, and mutual information. In information theory a Gaussian PDF model is adequate, because manmade signals can be designed to fit the selected parametric model and also because the channel noise statistics are basically Gaussian. Under these conditions Shannon entropy can be easily estimated. However, the problems and data in statistical signal processing and machine learning are often incompatible with this parametric methodology without raising the model bias issue. This is exactly where a nonparametric line of reasoning is helpful if an efficient and accurate PDF estimator can be derived.

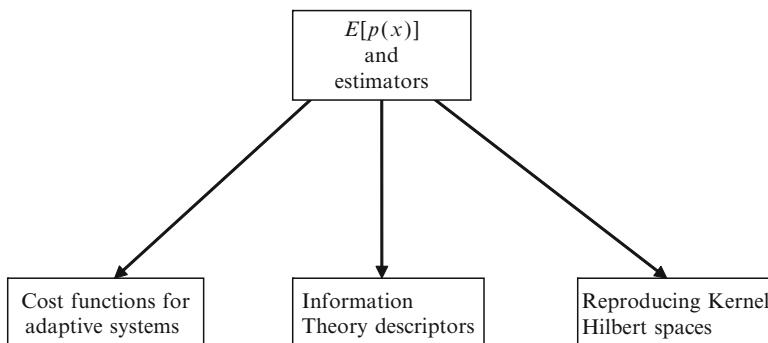
The author published in 2000 three major observations. First, in most engineering cases the error variable is a scalar or has low dimension, so PDF estimation is still accurate. Second, the requirements of a PDF estimator for an entropic cost function are very different from the ones for density estimation. Indeed, in adaptation what matters is to locate the extrema (maximum

or minimum) of a cost function, not the exact value of the cost. Therefore, as long as the free parameters of the entropic cost estimator are tuned to the data dynamic range and they are kept constant for the same data, relative comparisons of entropy are sufficient to find the optimal value of the system parameters. However, the estimator must be smooth to allow gradient descent searches. Third, although nonparametric PDF estimation for continuous random variables is difficult, the argument of the logarithm in Renyi's entropy  $(1 - \alpha)^{-1} \log \int p^\alpha(x) dx$  [264] is easier to estimate directly from data, in particular for  $\alpha = 2$ . Because  $E_X[p(x)] = \int p(x)p(x)dx$ , the  $\alpha = 2$  case is just the mean value of the data PDF. Moreover, the logarithm can be dropped from the cost function because optimal parameters are invariant to monotonic transformations on the cost. The core concept in this book became naturally  $E_X[p(x)]$ , which is called the information potential (IP) for reasons that become apparent later (Figure 1.2).

In simple terms, this book provides answers to the following questions. Under what conditions are ITL costs better than MSE to train a learning machine and how can they be practically implemented?

We called this framework *information-theoretic learning*, a terminology perhaps first used by Watanabe in the 1970s [330] and also coined by us in the context of adaptive systems training [252]. This terminology succinctly conveys the goals of this book. Namely, we seek to quantify global scalar descriptors of the underlying PDF (e.g., entropy), while being primarily interested in learning and adaptation. However, inasmuch as information is utilized extensively in our colloquial language and we use the mathematical theory of information developed initially by Claude Shannon and Alfred Renyi among many others, the inclusion of theoretic in the title is natural.

The bridge to reproducing kernel Hilbert spaces (RKHS) [35] in Figure 1.2 may appear as a surprise. It turns out that  $E_X[p(x)]$  is a special case of  $\int p(x)q(x)dx$  that occurs when two different PDFs  $p(x)$  and  $q(x)$  are equal. This integral is a bivariate positive definite function and, as we show, it



**Fig. 1.2.** Relationship of the information potential with the covered topics.

defines a linear space of functions (more precisely a Hilbert space) with finite evaluations called an RKHS. The argument of Renyi's quadratic cross-entropy,  $-\log \int p(x)q(x)dx$ , is called the cross information potential (CIP) so there is a straight relationship between information theory and the RKHS created by the CIP. RKHS are possibly infinite-dimensional linear spaces, and they are well studied and very important in physics, engineering, and machine learning because they possess an inner product structure to implement many of the algorithms required in these fields. They have been shown to provide a functional analysis view of Gaussian processes, which is very useful for many of the results addressed in this book and they also provide useful bridges to kernel learning. The CIP defines a RKHS for information-theoretic learning, extending the concept of entropy to functional spaces. This may be particularly useful in cases where the natural structure of the space of the random variables does not allow inner products (e.g., point processes).

No book loosely related to the mathematical theory of information can avoid starting with Claude Shannon's seminal work on communication theory and this is also our starting point. In Section 1.2 we show how the fundamental definitions of entropy and mutual information yielded a paradigm shift in the design of communication systems. Information theory (IT) has grown tremendously since Shannon, so we briefly present some generalizations of this work and the role of IT in machine learning.

In Section 1.3 we briefly review adaptive filter theory, which serves as the backdrop to explain the impact of entropic and divergence criteria in adaptive filters and pattern recognition applications. We present the gradient method for finding optimal parameters of linear systems, and describe the different types of learning and show how ITL can provide a unifying perspective for supervised and unsupervised learning from data that are normally treated independently.

In Section 1.4 of this chapter, we review reproducing kernels Hilbert spaces. A more systematic treatment of RKHS is provided later in the book, where we discuss the properties of the ITL estimators, present new similarity functions, design nonlinear filters, and compare ITL concepts with other efforts in kernel learning.

## 1.2 Information Theory

Information theory was conceptualized by Shannon [293] to deal with the problem of optimally transmitting messages over noisy channels. Although there is a physical substrate to communication systems (antennas, transmitters, receivers), the essence of information theory deals with characterizing the message structure and the limits of error-free transmission of the message's content, as its foundations are mathematical. Mathematical theories are rarely developed single-handedly and they normally take many years to be accepted and applied to practical problems. But information theory is

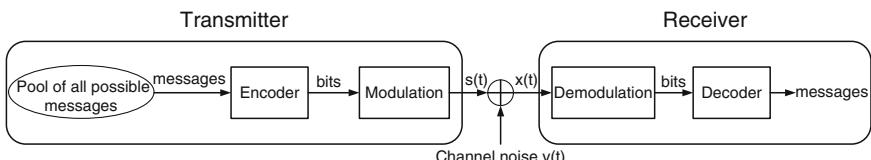
the brain-child of a single man, Claude Shannon, and was quickly accepted by the science and engineering communities. Information theory had an immediate impact in the design of communication systems, and it provided a mathematical framework to formulate and quantify interaction beyond physical laws, which is very important for society and in helping to understand the goal-driven behavior of biological organisms.

## Origins

The work of Norbert Wiener established that the best way to quantify the transmission of manmade signals over noisy environments involves a probabilistic framework both for the signals and the noise [333]. A signal  $s(n)$  can then be modeled as a random process, that is a sequence of random variables  $s_i$  over time with a given probability law which we assume constant across time (stationary random process). Wiener's idea of using a random basis to decompose even deterministic manmade signals was an important contribution that changed forever optimal signal processing.

For simplicity we start by describing binary messaging, random processes that can only take two discrete values, 0 or 1. The problem of communication over a noisy channel (Figure 1.3) can then be stated as follows. Starting from the transmitter side, a given message is drawn from the pool of all possible messages  $\{m_i\}$ . Next it is encoded to binary values and then translated by the transmitter into a physical symbol (voltage or current). Under a probability framework the chosen message is modeled as a Bernoulli process. These symbols are then corrupted by the channel noise  $v_n$ , which is normally modeled as stationary additive Gaussian noise  $v_i$  with a given variance (the power of the noise). At the receiver, a new sequence of symbols  $x_i = s_i + v_i$  is measured. The problem of the communication system designer is to find out how to recover  $s_i$  from  $x_i$ , making as few errors as possible.

It is useful to formulate the design of a communication system in abstract terms in order to elucidate the core problems. Optimum channel design constitutes a compromise among three factors: signal-to-noise ratio (SNR), message rate, and power. How transmission power affected SNR was well understood. Sampling theory quantified the requirements to recover signals coded in discrete alphabets. The characterization of pulse code modulation (PCM) and pulse position modulation (PPM) showed that it was possible to trade SNR



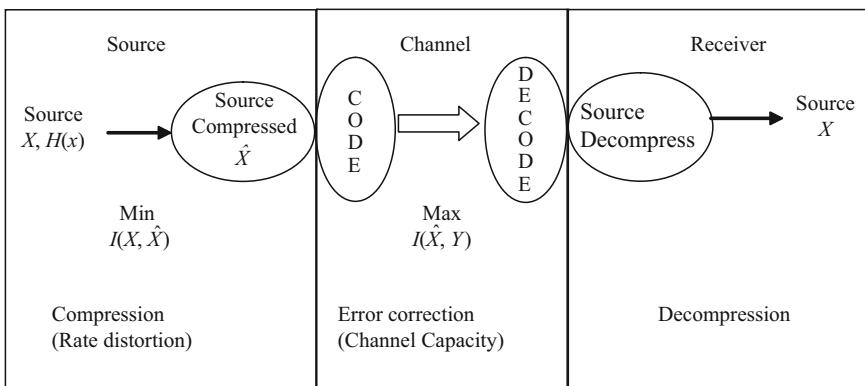
**Fig. 1.3.** Block diagram of a communication system.

for message rate. However, no one had the vision how to mathematically formulate the optimization problem, nor what the solution might be.

The strategy proposed by Shannon differed from that of Wiener. Wiener was the first to suggest attributing probabilities to messages, but he was more interested in prediction and control, so he proposed to optimally filter the noise from the received symbol sequence (the Wiener filter). Shannon, on the other hand, proposed a scheme where the transmitted signal should be modified first (predistorted or encoded) to withstand minimum degradation when transmitted through the channel and modified after (decoded) at the receiving end to regain the original message.

Information theory was exactly created to help study the theoretical issues of optimally encoding messages according to their statistical structure, selecting transmission rates according to the noise levels in the channel, and evaluating the minimal distortion in messages. Surprisingly, only two statistical descriptors are needed to accomplish this seemingly impossible task: *entropy* and *mutual information*, the latter of which is closely related to divergence (a dissimilarity measure) in probability spaces. According to Shannon [293], the optimal transmission of messages with a predetermined distortion over noisy channels is accomplished using the system shown in Figure 1.4: the probability structure of the messages dictates the optimal coding, as established in the *source coding theorem* (optimal coding exploits source redundancy, which is quantified by entropy). Practically speaking, when the message rate is too high for the channel one has to impose a message rate limit. The inclusion of a limit results in the need to optimally compress the messages, as established by the *rate distortion theory* (optimal compression is achieved by minimizing the mutual information between the original message and its compressed version).

In order to withstand the channel noise in the transmission, the source-compressed data are encoded for error-free transmission by maximizing the mutual information between the sent message and the received message, as



**Fig. 1.4.** An information-theoretic view of optimal communications.

established by the channel capacity theorem. Finally, the received message is decoded and decompressed to yield the original message (with the predefined distortion). There is a duality between the problem of data compression and data transmission in the sense that the first minimizes redundancy for efficiency, whereas the second adds redundancy to mitigate the noise effects in the channel. But what is remarkable is that the same quantity, mutual information, is specifying the two compromises for error-free communication: the data compression minimum limit (rate distortion) and the data transmission maximum limit (channel capacity).

### 1.3 Entropy

In 1928 Hartley reasoned that when one symbol is chosen from a finite set of possible symbols  $S$ , then the number of choices can be regarded as a measure of information and he called it the “amount of information” [137]. According to Hartley, the amount of information in a set of  $N$  transmitted symbols is  $H_0 = \log_{10} S^N = N \log_{10} S$ , i.e. proportional to the number of different choices (the base 10 logarithm is called the Hartley). Today we prefer to think in terms of two symbols  $[0,1]$  and the logarithm is normally taken as base 2 and the unit becomes the bit. One of the prerequisites for a measure of information is to be additive for independent events, so the logarithmic function is a “natural” measure, and it also handles rather well the physical systems that give rise to the set of events of interest in communications (i.e., an  $N$ -bit long digital word produces  $2^N$  different messages, so the amount of information would be  $N$ , the number of bits). Notice that no probabilistic reasoning was used by Hartley to derive the amount of information.

Today we can be more precise in the justification of the logarithm by defining a random variable  $x$  with a set of possible outcomes  $S_X = \{s_1, \dots, s_N\}$  having probabilities  $p_X = \{p_1, \dots, p_N\}$ , with  $p(x = s_k) = p_k, p_k \geq 0$  and  $\sum_{x \in S_x} p(x) = 1$ , and denote the number of elements in  $S$  by  $\#S$ . The number of *binary questions* guaranteed to identify any outcome in  $S_X$  is lower bounded by

$$I_H(X) = \log_2(\#S_X) \quad (1.1)$$

which is exactly Hartley’s amount of information. Moreover,  $H_0$  is additive for independent events, that is,

$$I_H(X, Y) = I_H(X) + I_H(Y). \quad (1.2)$$

It was Shannon who stated that one should go beyond the cardinality of the message set to accurately quantify how much choice is involved in the selection of probabilistic events when only the probabilities are known. The probability of selecting each message matters and should therefore be brought into the formulation. First, he noted that Hartley’s information content is only accurate if we do not know anything about the data, that is if we assume an

equal probability to all the events ( $p_k = 1/N$ ). In order to fully characterize an element of a set of symbols  $S_X$  occurring with different probabilities  $p_k$  the information amount of an outcome should be

$$I_k = \log_2 \frac{1}{p_k}. \quad (1.3)$$

The probabilistic view is that the amount of information for unequal probable messages is a measure of the unexpectedness in the data due to the inverse dependence on probability. In fact, for a single message, if  $p_k = 1$  then the information content of the message is zero (perfect knowledge), whereas if  $p_k$  is small, the message is unexpected so its information content is high. Therefore  $-\log p_k$  is a new random variable on  $S$  that is fully specified by the probability mass function (PMF) of the messages. We also use the notation  $I(x_k)$  to represent the amount of information. But if we are interested in the information conveyed by the random variable  $X$  defined on the set of messages  $S_X$ , how should one weight the information content? Shannon defined the *uncertainty* of the ensemble  $X$  as the sum across the set of the uncertainty in each message weighted by the probability of each message, or

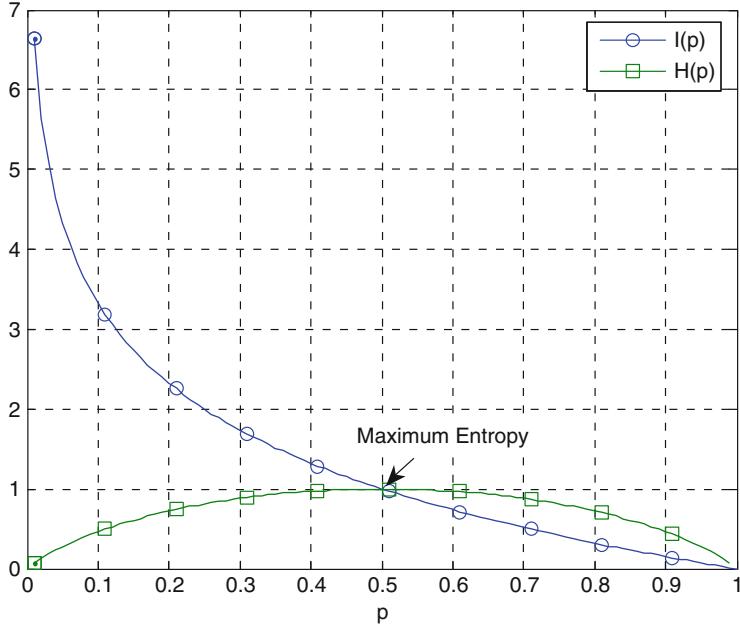
$$H(X) = - \sum_k p(x_k) \log_2 p(x_k) = -E[\log_2 p(X)]. \quad (1.4)$$

Shannon called his uncertainty measure *entropy* and it is measured in bits of information, with the assumption that for  $p(x_k) = 0$ ,  $p(x_k) \log p(x_k) = 0$ . This is the same expression as the same form of physical entropy, but information entropy is a property of the probability mass function, whereas physical entropy is a property of the state of the physical system. Notice that entropy measures the uncertainty in the data set using a single scalar quantity. It is also important to note that it is the combination of unexpectedness weighted by the probability that is really the essence of the concept of entropy: events that are very unlikely, and therefore of high information content, are discounted by their rare occurrence through the product with the probability. Likewise, events that are very likely have a low information content so that the product once again has a small value. This creates a “balance” that had hitherto never been quantified in probabilistic reasoning.

Figure 1.5 shows  $I(x_k)$  and  $H(X)$  for a Bernoulli random variable (which is uniquely defined by a single probability) for different values of probability  $p$ .

Notice also that  $H(X)$  depends on the shape of the distribution and  $H(X) \leq H_0(X)$ , where  $H_0(X)$  is the maximum entropy as shown in the figure. In words this states that not all random variables are equally random, and the scalar called entropy is able to characterize the uncertainty in the data, which is contained implicitly in the functional form of the PDF. The redundancy in  $X$  is exactly defined as  $R = (H_0(X) - H(X))/H_0(X)$ .

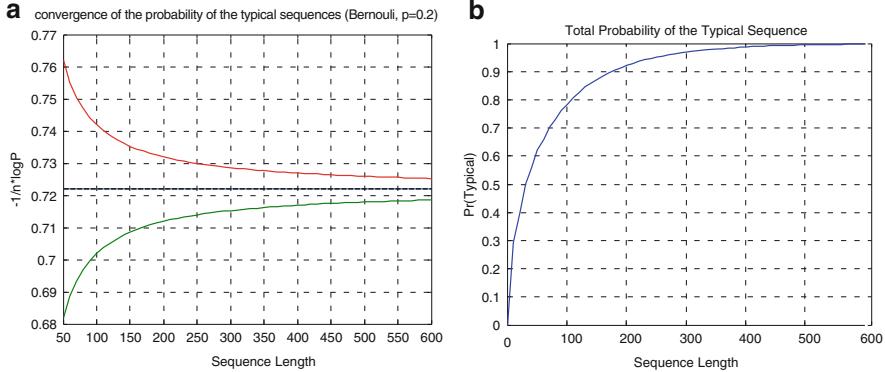
It is important to understand how a single scalar  $H(X)$  can produce a useful description of the PMF of the data, the latter of which is a function. The reason is because  $H(X)$  quantifies remarkably well the effective



**Fig. 1.5.** Information content versus entropy for Bernoulli random variable with probability  $p$  and  $1 - p$ .

“volume” spanned by the data in high-dimensional spaces. This basic result is contained in the *asymptotic equipartition property* (AEP) [65], which states the following. For an ensemble of  $N$  independent i.i.d. random variables,  $\mathbf{X} = (X_1, X_2, \dots, X_N)$  with  $N$  sufficiently large, the outcome  $\mathbf{x} = (x_1, x_2, \dots, x_N)$  is almost certain to belong to a subset of  $S_X^N$  having only  $2^{NH(X)}$  members, and all having probability close to  $2^{-NH(X)}$ . This means that there is a subset of elements of  $S$  (called the *typical set*) that specifies the probability of the set and describes the behavior of the distribution. Therefore, if  $H(X) << H_0(X)$  then  $2^{NH(X)}$  is a small fraction of the number of possible outcomes  $(\#S_X)^N = 2^{N H_0(X)}$ . The reason for this result is the fast convergence of  $-1/N \log p(X_1, \dots, X_N) \rightarrow H(X)$  for large  $N$  in all points of the domain except in the neighborhoods of 0 and 1. This result is conceptually similar to the law of large numbers, which justifies the use of the sample mean for approximating the expected value.

Suppose there is an i.i.d. Bernoulli random source with probability 0.2 and 0.8 for 0 and 1, respectively. If the length of the sequence is  $N$ , then the typical sequence should contain around  $0.2N$  zeros and around  $0.8N$  ones. Figure 1.6a illustrates the convergence of the probability across these sequences. Also, we can see from Figure 1.6b that the probability of the typical sequences defined above converges to 1 as the length increases.



**Fig. 1.6.** (a) Convergence of the typical sequences and; (b) the total probability as a function of  $N$ .

Entropy is not the only scalar descriptor of the probability mass (or density) function of the data, but it is arguably the most fundamental [264]. Still today, the most commonly used descriptors of PMFs (or PDFs) are the moments of the data [235]. For instance, the mean, which locates the center of mass of the PMF, is defined as

$$m(X) = E[x] \cong \frac{1}{N} \sum_k x_k$$

and the variance, which measures the dispersion around the mean, is defined as

$$\text{var}(X) = E[(x - m)^2] \cong \frac{1}{N} \sum_k (x_k - m)^2.$$

Likewise, the  $i$ th moment around the mean is  $m_i(X) = E[(x - m)^i]$ . For  $i = 3$  the moment is referred to as the skewness, which measures the asymmetry in the PMF, whereas  $i = 4$  is referred to as the kurtosis, which measures the peakness with respect to the Gaussian PMF. As is well known, the PMF can be entirely defined by the central moment expansion of the data (or alternatively by the characteristic function  $\varphi_X(s) = E_X[e^{jsX}]$ ), but one needs an infinite set of such numbers.

The definition of entropy is fundamental because of its elegant properties, deep meaning, and above all, because it enabled numerous developments in communication theory, which brought a paradigm shift to the design of optimal communication systems. In terms of elegance, Shannon proved that his definition of entropy (for discrete random variables and apart from a scale factor that is related to the units of measurement) was the only one that obeyed a very simple and intuitive set of axioms [293]:

1.  $H(p, 1-p)$  is a continuous function of  $p$ .
2.  $H(p_1, p_2, \dots, p_N)$  is a symmetric function of the  $p_k$ .

3.  $H(1/n, 1/n, \dots, 1/n)$  is a monotonically increasing function of  $n$ .
4.  $H(p_1, p_2, \dots, p_N) = H(p_1 + p_2, p_3, \dots, p_N) + (p_1 + p_2)H(p_1/(p_1 + p_2), p_2/(p_1 + p_2))$ .
5.  $H(p_1, p_2) = H(p_1) + H(p_2)$  for independent events.

The fourth property is called recursivity and it singles out Shannon's entropy from all the other definitions of entropy. There is also a sixth property known as permutational symmetry, which was implicitly used by Shannon. Entropy is a concave function of its arguments [65], therefore a large body of mathematics can be readily applied. Perhaps more important than this axiomatic characterization, Shannon was able to build a mathematical theory for the information content of messages and prove three very important theorems: the source coding theorem, the channel capacity (or channel coding) theorem, and the rate distortion theorem (see [65])

## 1.4 Mutual Information

Up to now we dealt with the characterization of a single source of information. However, communication systems have inputs and outputs, each uncertain in its own way, therefore it is important to expand the analysis to two sources of information. This expansion leads us to the second important descriptor of statistical properties used in information theory, which Shannon named *mutual information*. Let us create a discrete product space of the transmitted message  $X = \{x_k\}_{k=1}^N$  and the received message  $Y = \{y_k\}_{k=1}^N$  in Figure 1.3. The channel noise creates a probability distribution  $p(X, Y)$  over the product space. From the point of view of error-free communication in the receiver, the question is to what extent does  $y_i$  specify one of the  $x_k$ . The channel basically changes the probability of  $x_k$  from its a priori value  $p(x_k)$  to the a posteriori value  $p(x_k|y_i)$ . This is just plain old probabilistic theory. We previously discussed the characterization of the information content of  $X$  by its entropy, now we would like to determine how much information can be communicated through the channel. After observing  $y_i$  the probability of  $x_k$  is denoted  $p(x_k|y_i)$  and the uncertainty left in  $x_k$  becomes  $\log(1/p(x_k|y_i))$ . Therefore, the decrease in uncertainty about  $x_k$  brought about by observing  $y_i$  is

$$I(x_k, y_i) \equiv \log_2 \left( \frac{1}{p(x_k)} \right) - \log_2 \left( \frac{1}{p(x_k|y_i)} \right) = \log_2 \frac{p(x_k|y_i)}{p(x_k)} = \log_2 \frac{p(x_k, y_i)}{p(x_k)p(y_i)} \quad (1.5)$$

which can complementarily be thought of as the gain in information. Notice that if  $p(x_k) = p(x_k|y_i)$ ,  $I(x_k, y_i) = 0$ , which indicates that there is no decrease in uncertainty. In the other extreme, if  $p(x_k|y_i) = 1$  then  $I(x_k, y_i) = \log(1/p(x_k))$ ; that is all the information about  $x_k$  was conveyed through the

channel. For some messages the gain of information can be positive, whereas it can be negative for others. Notice also that  $I(x_k, y_i) = I(y_i, x_k)$ . The total decrease in uncertainty in  $X$  by observing  $Y$  is known as the mutual information between  $X$  and  $Y$ , which is defined as

$$\begin{aligned} I(X, Y) &= E[I(x_k, y_i)] = \sum_i \sum_k p(x_k, y_i) \log_2 \frac{p(x_k | y_i)}{p(x_k)} \\ &= \sum_i \sum_k P(x_k, y_i) \log_2 \frac{p(x_k, y_i)}{p(x_k)p(y_i)} \end{aligned} \quad (1.6)$$

where  $p(X, Y)$  is the joint mass function of  $X$  and  $Y$ .

In communications, the base of the logarithm is normally two so that the units are measured in bits, but any other basis can be used. It is appropriate to relate mutual information with the probability theory concepts of joint and condition PMFs. One can start by defining the joint entropy of a pair of random variables  $X$  and  $Y$  as

$$H(X, Y) = - \sum_x \sum_y p(x, y) \log p(x, y) = -E_{X,Y}[\log p(X, Y)]. \quad (1.7)$$

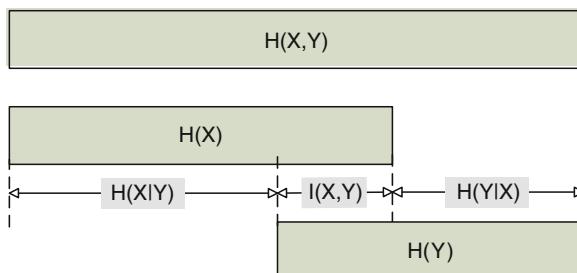
Likewise we can define the conditional entropy of  $Y$  given  $X$  as

$$H(Y|X) = - \sum_x \sum_y p(x, y) \log p(y|x) = -E_{X,Y}[\log p(y|x)]. \quad (1.8)$$

Therefore, mutual information can also be formally defined by expanding Eq. (1.6) as

$$\begin{aligned} I(X, Y) &= H(X) + H(Y) - H(X, Y) \\ I(X, Y) &= H(X) - H(X|Y) = H(Y) - H(Y|X) \end{aligned} \quad (1.9)$$

The relationships between mutual information and these quantities can be visually appreciated on a Venn diagram, as shown in Figure 1.7.



**Fig. 1.7.** Diagram to illustrate the relationship among joint information, marginal entropy, conditional entropy, and mutual information.

As one can see, mutual information is always greater than zero and it quantifies the intersection of  $H(X)$  and  $H(Y)$ . From a communication system perspective and in words, mutual information quantifies the reduction of uncertainty in  $X$  after observing  $Y$ , that is  $H(X) - H(X|Y)$ . But sometimes it is easier to perform the computation in reverse order, that is computing  $H(Y) - H(Y|X)$ , from which the same result will be obtained.

## 1.5 Relative Entropy and Kullback–Leibler Divergence

In 1936, Mahalanobis first introduced the concept of “distance” between two probability distributions, and since then many important results related to this concept have been established. Let us consider two different probability densities  $p(x)$  and  $q(x)$ , and define the Kullback–Leibler (KL) divergence as

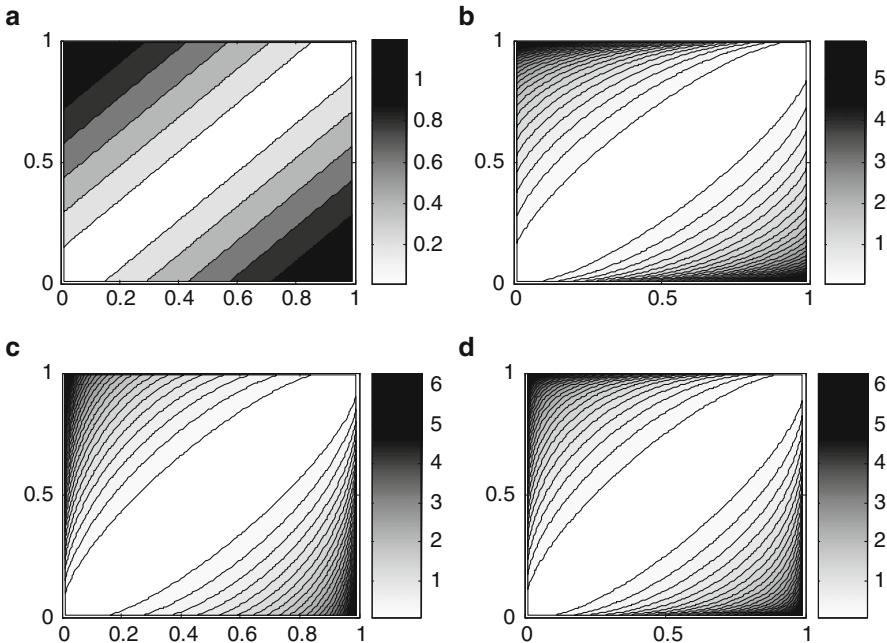
$$D_{KL}(p \parallel q) = \sum_x p(x) \log \frac{p(x)}{q(x)} = E_p \left[ \log \frac{p(X)}{q(X)} \right]. \quad (1.10)$$

The KL divergence is effectively measuring dissimilarity (which conceptually is a distance) between  $p(x)$  and  $q(x)$ . However, we cannot call it a distance because it only obeys one of the postulates of distance. More specifically, the KL divergence obeys the positivity requirement ( $D_{KL}(p||q)$  is nonnegative), but it is not symmetric ( $D_{KL}(p||q)$  differs from  $D_{KL}(q||p)$  in general), and it does not obey the triangular inequality. For this reason it is called a directed distance or a *divergence*. The quantity  $-E_p[\log q(X)]$  is commonly named *cross entropy*. Figure 1.8 shows the contours of constant value of dissimilarity over the  $p(x)$  and  $q(x)$  space, obtained with the conventional Euclidean distance for a pair of distributions ( $D_{ED}(p, q)$ ),  $D_{KL}(p||q)$ ,  $(D_{KL}(q||p)$ , and  $J_{div}(p||q)$  respectively (the latter is defined shortly). As we can see the assessment of dissimilarity is quite different for the four measures.

The asymmetry may be a useful property when one addresses directional coupling, but many times one would like to work with an information dissimilarity measure that is symmetric. Jeffrey [65] defined a symmetric measure that is now known as *J divergence*, which is given as

$$J_{div}(p, q) = \sqrt{1/2(D_{KL}(p \parallel q))^2 + 1/2(D_{KL}(q \parallel p))^2}. \quad (1.11)$$

There is kind of Pythagorean theorem that supports the interpretation of the KL divergence as square distance. Suppose that the distribution  $p$  lies in a convex set  $S$  and  $Q$  lies outside  $S$ . Choose a distribution  $p^*$  on the boundary of the convex set such that  $D_{KL}(p^*||q)$  assumes its minimum value for  $p^*$  belonging to  $S$  and fixed  $q$ . It is possible to show [178] that  $D_{KL}(p \parallel q) \geq D_{KL}(p \parallel p^*) + D_{KL}(p^* \parallel q)$ . This agrees with the Euclidean distance between the same points  $D_E^2(p, q) \geq D_E^2(p, p^*) + D_E^2(p^*, q)$ .



**Fig. 1.8.** Measures of distinguishability between probability distributions  $P = (p, 1 - p)$  and  $Q = (q, 1 - q)$ : (a)  $D_E(P||Q)$ , (b)  $D_{KL}(P||Q)$  (c)  $D_{KL}(Q||P)$  (d)  $J_{div}(P, Q)$ .

### Properties of KL Divergence

So far we have related the concept of entropy with uncertainty. It seems reasonable that reduction of entropy should result in a gain of information but we have not provided a systematic treatment of this aspect. The KL divergence is associated with the concept of gain of information, which is considered by many as the fundamental concept in information theory [264].

Let the probability of an event  $A$  be  $P(A) = q$  and let it change to  $p$  after observing an event  $B$ ,  $P(A|B) = p$ . We obtain  $\log 1/q - \log 1/p$  bits of information on  $A$  by observing  $B$ . Likewise, using the concept of entropy, we could say that the uncertainty in  $A$  was  $\log 1/q$  and it changed to  $\log 1/p$  after observing  $B$ . The decrease in uncertainty is again given by  $\log 1/q - \log 1/p$  bits. Now, if one has events  $A_1, \dots, A_N$  with  $q_k = P(A_k)$ ,  $k = 1, \dots, N$ , when observing  $B$  these probabilities become  $p_k = P(A_k|B)$ . How much information was gained by observing  $B$  in this case? The decrease in uncertainty (gain in information)  $\log p_k/q_k$  is positive for some and negative for others. The total gain in information can be calculated in two different ways: by taking the average of the partial gains of information or by averaging the negated increase in partial uncertainty, multiplied by  $p_k$ , which yields  $D_{KL}(P||Q)$  or the *relative entropy*. This quantity is also called *Shannon's information gain*.

and it is exactly the decrease in uncertainty, which distinguishes Shannon's from all the other information measures defined in the literature. For the particular case that  $q(x)$  is the uniform distribution, the relative entropy defaults to the entropy of  $p(x)$ .

Mutual information is a special case of the KL divergence, which is obtained when  $p(x)$  is the joint PDF of  $x$   $y$ , and  $q(x)$  is the product of the marginals:

$$D_{KL}(p(X, Y) \parallel p(X)q(Y)) = I(X, Y). \quad (1.12)$$

The relative entropy is jointly convex; that is for any  $\lambda \in [0, 1]$ ,

$$D_{KL}(\lambda p_1 + (1 - \lambda)p_2 \parallel \lambda q_1 + (1 - \lambda)q_2) \leq \lambda D_{KL}(p_1 \parallel q_1) + (1 - \lambda)D_{KL}(p_2 \parallel q_2)$$

When applied to Markov chains, KL divergence has a very distinct property that is very useful. Let  $T$  be a stochastic map, i.e. a properly normalized matrix that transforms one probability distribution into another. Then we can prove [65] that

$$D_{KL}(Tp(X) \parallel Tp(Y)) \leq D_{KL}(p(X) \parallel p(Y)) \quad (1.13)$$

or in words, that the relative entropy in a Markov chain always decreases with time.

Another property of KL divergence that is worth stating in this brief review is that it is invariant to a reparameterization of the random variable. This is very important and it is a reflection that there is a close relationship between the KL divergence and the local structure of the space quantified by the Fisher–Rao metric [178].

## Continuous Variables

Up to now we have concentrated on discrete event spaces, but it is possible to extend the definitions of mutual information and entropy to continuous event spaces. The (differential) entropy of a continuous random variable  $X$  with PDF  $p(x)$  is defined as  $h(X) = -\int_S p(x) \log p(x) dx$ , provided the integral exists and where  $S$  is the support of the random variable  $X$ . Although in this book we are interested in continuous random variables, we use the notation  $H(X)$  to represent differential entropy and also refer to it simply as entropy. Therefore entropy and mutual information for continuous random variables are defined as

$$H(X) = - \int \log p(x)p(x) dx = -E_X[\log(p(x))], \quad (1.14)$$

$$I(X, Y) = \iint I(x, y)p(x, y) dx dy = E_{X,Y}[I(x, y)], \quad (1.15)$$

$$D_{KL}(p(x) \parallel q(x)) = \int p(x) \log \frac{p(x)}{q(x)} dx = E_p \left[ \log \frac{p(x)}{q(x)} \right], \quad (1.16)$$

where  $p(\cdot)$  and  $q(\cdot)$  are PDFs,  $I(x, y)$  is given by Eq. (1.5) and  $E[\cdot]$  is the expected value operator. However, we must remember two things: entropy for the continuous variable case is not bounded from below (i.e., the entropy of the delta function, or a sum of delta functions, distribution is minus infinity); and entropy is not invariant to coordinate transformations.

## 1.6 Information Theory beyond Communications

After the pioneering work of Shannon, there was an immediate interest in better understanding the properties and applications of entropy and divergence. Information theory became a scientific field in itself and many mathematicians expanded upon Shannon's fundamental concepts. Moreover, information theory was also used in physics, statistics, and biology as well as in other areas of engineering such as signal processing and machine learning. This, book uses entropy and divergence as similarity metrics for optimization, thus it is important to briefly review some of these developments to fully appreciate the power and generality of these concepts.

### Generalized Definitions of Entropy and Divergences

The original definitions of entropy and divergence have been extended in numerous different directions, and today they can be considered as descriptors for a large class of concave functions. Burbea and Rao [43] introduced the  $\phi$ -entropy defined as

$$H_\phi(X) = \int \phi(p(x))dx, \quad (1.17)$$

where  $p(x)$  is the PDF of the continuous random variable  $x$ ,  $\phi$  is a continuous concave real function over the positive reals, and  $\phi(0) = \lim_{t \rightarrow 0^-} \phi(t)$ . There were some important definitions of entropy that could not be written in this form, therefore Salicru [278] defined the  $(h, \phi)$  entropies as

$$H_\phi^h(X) = h \left( \int \phi(p(x))dx \right), \quad (1.18)$$

where  $\phi$  is a continuous concave (convex) real function and  $h$  is a differentiable and increasing (decreasing) real function. Most of the definitions of entropy presented in the last 50 years can be written as  $(h, \phi)$  entropies. Some examples are presented in Table 1.1.

Likewise, Csiszar [66] defined the  $\phi$  divergences between the probability densities  $p(x)$  and  $q(x)$  as

$$D_\phi(p(x), q(x)) = \int q(x)\phi\left(\frac{p(x)}{q(x)}\right)dx, \quad (1.19)$$

**Table 1.1.** Some Entropies Written as  $(h, \phi)$  Entropies

Entropy type	$\phi(x)$	$h(x)$
Shannon	$-x \log(x)$	$x$
Renyi	$x^\alpha$	$[\alpha(1 - \alpha)]^{-1} \log(x)$
Havrda–Charvat (Tsallis)	$(1 - \beta)^{-1}(x^\beta - x)$	$x$

**Table 1.2.** Divergences Written as  $(h, \phi)$  Divergences

Divergences	$\phi(x)$	$h(x)$
Kullback-Liebler	$x \log(x) - x + 1$	$x$
Renyi	$(x^\alpha - \alpha(x - 1) - 1)/(\alpha(\alpha - 1))$	$(\log(\alpha(\alpha - 1)x + 1)/(\alpha(\alpha - 1))$
Bhattacharyya	$-x^{1/2} + 0.5(x + 1)$	$-\log(-x + 1)$
J-Divergence	$(x - 1) \log(x)$	$x$

where  $\phi$  belongs to the class of convex functions on the positive real plane, such that at  $x = 1$   $\phi(1) = 0$  and at  $x = 0$ ,  $\phi(0) = 0$ . A special case of the  $\phi$  divergence is the  $\alpha$  divergence ( $\alpha$  real value) defined as  $D^\alpha = D_{\phi(\alpha)}$  obtained as

$$D^\alpha(p(x), q(x)) = \frac{1}{\alpha - 1} \int q(x) \left( \frac{p(x)}{q(x)} \right)^\alpha dx \quad \alpha \neq 1 \quad (1.20)$$

and which directly gives the KL for  $\alpha \rightarrow 1$ , the Hellinger for  $\alpha = 1/2$  and Renyi's  $\alpha$  entropy when the density  $q(x)$  dominates  $p(x)$  and is uniform over the domain. As with the entropies there were important divergences not considered in this definition, therefore Menendez et al [216] proposed the  $(h, \phi)$  divergences defined as

$$D_\phi^h(p(x), q(x)) = h(D_\phi(p(x), q(x))), \quad (1.21)$$

where  $h$  is a differentiable increasing real function defined on  $[0, \phi(0) + \lim_{t \rightarrow \infty} \phi(t)/t]$ . Table 1.2 shows some of the more conventional divergences expressed as  $(h, \phi)$  divergences.

The work on entropies and divergences is still flourishing and there are many new exciting ideas being researched today that may have a potential large impact in learning theory. For instance, an important divergence that escapes this taxonomy is the Bregman divergence [41] which is defined for strictly convex and differentiable real functions  $f(x)$  between two points of the domain  $P$  and  $Q$  as

$$B_f(P||Q) = f(P) - f(Q) + (P - Q) \cdot \nabla f(Q), \quad (1.22)$$

where  $\nabla$  is the partial derivative operator. The Bregman divergence can be interpreted as the distance between  $f(P)$  and the first-order Taylor expansion of  $f$  at  $Q$ , evaluated at  $P$ . Bregman divergence is a generalization of the

Euclidean distance ( $f(x) = \|x\|^2$ ), and it is also related to the KL divergence ( $f(p) = \sum_k p_k \log p_k - \sum_k p_k$ ).

Let us briefly examine Renyi's parametric family of entropies which plays a central role in this book and is given by

$$H_\alpha(X) = \frac{1}{1-\alpha} \log \sum_k p^\alpha(x_k) \quad \text{or} \quad H_\alpha(X) = \frac{1}{1-\alpha} \log \int p^\alpha(x) dx, \quad \alpha \geq 1$$

for discrete and continuous random variables, respectively, where  $\alpha$  is a free parameter. One of the advantages of Renyi's definition of entropy for estimation is that the logarithm appears outside the sum (or the integral). This has two important implications in our studies. First, it helps separate the effect of the logarithm, which is required to fulfill the additivity of the independent events property, from the role of the argument as a mesoscopic descriptor of random variables. Indeed,  $\alpha = 2$  is Renyi's quadratic entropy, which yields  $H_\alpha(X) = -\log \sum_k p^2(x_k) = -\log E[p(X)]$ . Notice that the argument of the logarithm is the information potential and is exactly the focal point of Figure 1.2 linking three seemingly unrelated topics. Moreover, we show that  $\alpha \rightarrow 1$  yields Shannon entropy. Second, it opens up the possibility of applying ITL to signal processing and machine learning problems even when the user does not have a clue about the PDF of the experimental data. In such cases a parametric PDF model is not recommended. This book utilizes Renyi's instead of Shannon's entropy as a starting point for ITL, primarily because the information potential can be estimated nonparametrically from pairwise sample differences. In adaptation, one can directly use the information potential instead of Renyi's entropy for several different criteria. For example, we derive cost functions for adaptation, as well as estimators for divergences and mutual information, all based on the information potential and its estimator.

## Information Theory and Machine Learning

One of the key contributions of information theory for probabilistic reasoning is to specify in unambiguous terms the amount of information (the information measure) that an observer possesses concerning a given phenomenon when only the PDF is known. This is of exceptional value for the design of communication systems because entropy is a scalar descriptor that abstracts an important PDF quantifier and it is sufficient to specify optimal design goals. Therefore, entropy and mutual information should be included in the class of mesoscopic data descriptors, on par with the moment decompositions. The following two questions naturally arise. For what class of learning problems are entropy and mutual information acceptable mesoscopic descriptors of the data? Can they be applied even in the case that the processing goal does not involve information measures?

In order to answer these questions, it is important to contrast the realities of information theory with machine learning. First and foremost, machine

learning and advanced signal processing applications can not be dependent upon parametric models of the data PDF to estimate entropy and mutual information. Indeed, the real world data sets and the diversity of applications are conducive to nonGaussian PDFs that can even change in time. Therefore, the first conclusion is that the usefulness of IT for machine learning is predicated upon non-parametric estimators of entropy and mutual information.

Second, one must be cognizant of the differences between IT applications and machine learning problems. As summarized in Section 1.2, to create optimal source codes, optimal message compression and take full advantage of the channel capacity, communication theory assumes full knowledge of the message statistics and additive channel noise. It turns out that the fundamental problem in machine learning is rather different. In unsupervised learning, the input data statistics are not only unknown but their quantification constitutes the goal of the analysis. Likewise, in supervised learning the goal is to estimate the joint PDF between the inputs and the desired response. This means that the a priori knowledge required to apply information theory to the optimal design of communication systems turns out to be the main final objective in many machine learning settings. From this perspective, it is not even clear that information theory will play a role in machine learning research.

Indeed many authors have experienced difficulties in applying information theory concepts in the biological sciences, and there is an absence of success stories that mimic the drastic impact that IT had in the development of communication theory [171]. Let us give a simple example in autonomous robotics. The goal in autonomous robotics is to model the “world”, that is, a given workplace to implement goal-directed behavior. Let us assume that the system is adaptive and it learns from its inputs with an information principle given by the log-likelihood of the data. The problem is that once an event is learned the machine state changes, but the probabilities in the external world remain the same. This is similar to what happens in the cognitive sciences, where objective probabilities cannot represent individual knowledge or beliefs nor can they discriminate between events that are of great importance or simply irrelevant for an individual. A productive approach is to first understand how to fully exploit conditional information for autonomous robotics before attempting to assign meaning to information.

Even with these concerns, this author believes that there are great advantages of using an IT framework for machine learning even when information measures are not involved in the solution. From our vintage point, the primary niches for IT applications in machine learning are to:

- Develop new frameworks for the design of learning systems
- Foster new cost functions for optimization and learning algorithms.
- Quantify data better with mesoscopic descriptors

Information theory has a lot to offer in helping create new paradigms for the optimal design of complex engineering systems because it focus the

designer's attention in the preservation and transfer of information among the sub-systems. As an example, we provide the IT perspective of the fundamental dilemma in machine learning: the trade-off between model complexity and generalization accuracy. Most readers have heard about the Occam's razor principle and the many, many alternate ways of framing and proposing solutions to this problem. According to Tishby et al [316], information theory can provide an alternative view and a new set of tools because of the abstract and principled concept of mutual information. It is not difficult to bridge rate distortion and compression theories with the complexity–accuracy dilemma. In fact, the rate distortion theorem specifies the simplest representation  $\hat{X}$  (in bits/s) of the input  $X$  for a given expected accuracy by solving

$$R(D) = \min_{d \leq D} I(X, \hat{X}) \quad (1.23)$$

with  $D$  being the distortion, whereas the channel capacity theorem specifies the best prediction  $\hat{X}$  (maximum number of bits/s) that can be sent reliably for a given condition of the channel (such as power) by solving

$$C(E) = \max_{e \leq E} I(\hat{X}, Y). \quad (1.24)$$

Therefore the recipe for using IT to address the complexity–accuracy dilemma is the solution of a MIN–MAX problem that involves mutual information between two pairs of variables. First, the input data  $X$  should be optimally compressed by a model and then its output  $\hat{X}$  should transfer as much information as possible with respect to the desired data  $Y$ . According to Linsker [199], biology operates an amazing compression within the sensory cortices to deliver features to the rest of the brain by maximizing exactly the information transfer across deep networks. Perhaps this is nature's way of coping with the curse of dimensionality, one of the unsolved problems in experimental science. Tishby et al. have proposed the information bottleneck method [316] as a self-organizing principle using mutual information to optimally compress input  $X$  (very much like Linsker's principle), and optimally transfer information to a second variable  $Y$  using the compressed data input. We also propose in Chapter 8 a novel statistical inference paradigm based on a multiobjective cost function combining entropy and Kullback–Leibler divergence, which directly yields the most common unsupervised learning applications (clustering, principal curves, and vector quantization).

This book mostly discusses the second bullet, IT-based cost functions for learning. We show how Renyi's entropy, which through its special form decouples the logarithm from its argument, has opened up new applications for optimal filtering, regression, clustering, and classification, all based on the exploitation of the 2-norm of the data PDF as a cost function. Therefore we envisage the data mean (as a location parameter) and the 2-norm of the PDF as alternate mesoscopic descriptors to the conventional least squares cost functions. We show in Chapters 10 and 11 that ITL ideas even lead to the development of novel mesoscopic similarity functions.

Finally, entropy and mutual information quantify more precisely the data's statistical microstructure when compared with second order statistics which still constitute the mainstream of statistical signal processing and machine learning using mesoscopic models. As we are going to see, ITL replaces 2nd order moments with a geometric statistical interpretation of data in functional spaces where variance is substituted by entropy, correlation by correntropy, mean square error (MSE) by Minimum error entropy (MEE), and distances in data space by distances in probability spaces.

In reverse direction, we also expect that learning theory will contribute to information theory, primarily by the development of Hilbert spaces where the inner product structure is related to the descriptors of entropy and divergence, even in cases where the structure of the original data does not support such operations (e.g., point processes). The definition of a RKHS for ITL operators is studied in Chapter 9.

## 1.7 Adaptive Model Building

Wiener and Kolmogorov's framework to seek optimal projections in spaces defined by stochastic processes initiated modern optimal filtering and changed forever our thinking about signal processing [185, 333]. The roots of adaptive model building go even farther back to the nineteenth century, when scientists started describing real data by linear relationships and correlations between independent variables. The combination of the Gaussian assumption and second-order statistical criteria has withstood the test of time and has led to mathematically convenient and analytically tractable optimal solutions, which could be easily studied through conventional calculus, linear algebra, and functional analysis. The most familiar examples are the mean square error in least squares linear regression, and output variance in principal components analysis (PCA).

The potential of optimal filtering became fully realized with the advent of digital computers, when the Wiener solution could be solved analytically for FIR filters using least-square algorithms. Adaptive methodologies that search for the optimal solution very efficiently such as Widrow's least mean square (LMS) [332] could be implemented in digital signal processors to solve optimally (in the MSE sense) and in real-time challenging signal-processing tasks. A curiosity at first, stochastic adaptive algorithms (i.e., processing the incoming data samples on a one-by-one basis) have become pervasive in signal processing and machine learning because they can be applied to problems where analytical solutions do not exist, as in the case of nonlinear filters. A noteworthy example is the backpropagation algorithm from neural networks [331].

In adaptive systems research (which is broadly used here to encompass traditional adaptive filtering as well as neural networks and various branches of machine learning), the user starts by specifying a parametric mapper (a projector or a filter), which can be linear or nonlinear, an adaptation algorithm

for the parameters (weights), and a criterion for optimality (see Figure 1.1). Supervised learning algorithms traditionally use the mean square error criterion defined as  $E[e^2(n)]$  (where  $e(n) = z(n) - y(n)$  is the error sequence) as the figure of merit in adaptation, which represents the sufficient statistics for the case of zero-mean signals and linear systems under Gaussian residual error assumptions [141, 333]. The emphasis on second-order statistics as the choice of optimality criterion is still prevalent today. This is understandable for the three main reasons:

1. The success of linear systems combined with second-order statistics, at least in part due to the inevitable Gaussianization effect of convolution
2. The well established mathematical framework
3. The abundance of efficient adaptive algorithms

Although the combination of second order-statistics and the Gaussianity assumption, the latter of which is supported by the central limit theorem, provide successful engineering solutions to most practical problems, it has become evident that this approach does not always work well. For example, it often has trouble adapting nonlinear systems and observation noise having fat-tail distributions or outliers [141]. Therefore, criteria that not only consider the second-order statistics but that also take into account the higher-order statistical behavior of the systems and signals are desired. Recent work in both the control literature [99] and the signal processing/machine learning literature [51, 95, 102] addresses this issue. For instance, in blind separation of sources and blind deconvolution of linear channels, the insufficiency of second-order statistics in stationary environments has led to new approaches incorporating higher order statistics into adaptation. Specifically, the field of independent components analysis (ICA) has benefited greatly from the use of information-theoretic performance measures [156].

## From Linear Adaptive to Information Filtering

In ITL we use information-based criteria to adapt systems. For example, several chapters of this book are devoted to training linear and nonlinear systems using the entropy of the error signal. Hence it is appropriate to refer to this process as *adaptive information filtering*. Likewise, because optimal filtering of stochastic processes is essentially regression for multivariate random variables, the methods can be applied without modification to information regression.

Training utilizes sensitivity considerations to optimize the system's parameters during adaptation (Figure 1.1). Let us consider a single-output parametric system  $y = f(x, w)$  and a set of training input desired response pairs  $\{x(n), z(n)\}$  where  $n$  is a time index. The system output  $y(n)$  is compared with the desired response  $z(n)$  at each time step and an error is defined as  $e(n) = z(n) - y(n)$ . We start by assuming that  $f(\cdot)$  is a linear function

of  $x(n)$  and the parameters, such as in finite impulse response (FIR) filters  $y(n) = \mathbf{w}^T \mathbf{x}(n)$  or equivalently

$$y(n) = \sum_{k=0}^{M-1} w_k x(n-k), \quad (1.25)$$

where  $\mathbf{x} = [x(n), \dots, x(n-M+1)]^T$ ,  $\mathbf{w} = [w_0, \dots, w_{M-1}]^T$  are the FIR coefficient (also called the weights) and  $M$  is the filter order. For future reference, we sometimes denote  $x(n-k) = x_k(n)$  to represent the value at tap  $k$  and time  $n$ . Let  $J(e(n))$  be the cost function constrained to be a continuous function of the error. The conventional adaptive filtering framework [141] describes the optimal filtering problem as one of obtaining the minimal error in the mean square error sense between the desired response  $z(n)$  and the system output  $y(n)$ :

$$J_w(e(n)) = E[(z(n) - f(x(n), \mathbf{w}))^2]. \quad (1.26)$$

The general approach to finding a set of parameters that correspond to a stationary point of  $J(e(n))$  is to take the partial derivatives with respect to the unknowns (in this case the system parameters  $\mathbf{w}$  also called the weights) and equate them to zero; that is,

$$\frac{\partial J(e(n))}{\partial \mathbf{w}} = 0. \quad (1.27)$$

Using the chain rule (see the block diagram of Figure 1.1), we immediately obtain

$$\frac{\partial J(e(n))}{\partial \mathbf{w}} = \frac{\partial J(e(n))}{\partial e(n)} \frac{\partial e(n)}{\partial \mathbf{w}} = 0. \quad (1.28)$$

The first term on the right side is calculated from the cost function, whereas the second term is the sensitivity calculated across the specified system topology. For an FIR filter and the mean square error cost this yields

$$\frac{\partial J(e(n))}{\partial \mathbf{w}} = E \left[ \frac{\partial e^2(n)}{\partial e(n)} \frac{\partial e(n)}{\partial \mathbf{w}} \right] = -2E[e(n)\mathbf{x}(n)] = 0. \quad (1.29)$$

When interpreted in vector spaces this expression states that the optimal solution occurs when the error is orthogonal to the input space. If we substitute the definitions of  $e(n)$  in Eq. (1.29) we obtain [185]

$$\begin{aligned} E[(z(n) - y(n))x(n-i)] &= E[z(n)x(n-i)] \\ -E \left[ \sum_{k=0}^{M-1} w_k x(n-k)x(n-i) \right] &= 0, \quad i = 0, \dots, M-1. \end{aligned} \quad (1.30)$$

The solution of this set of  $M$  equations in  $M$  unknowns yields the length- $M$  vector of weights  $\mathbf{w} = \mathbf{R}^{-1}\mathbf{p}$  where  $\mathbf{R}$  is the  $(M \times M)$  autocorrelation matrix of the input  $x(n)$  defined as

$$\mathbf{R} = \begin{bmatrix} R_{0,0} & \cdots & R_{0,M-1} \\ \cdots & \cdots & \cdots \\ R_{M-1,0} & \cdots & R_{M-1,M-1} \end{bmatrix}, \quad R_{k,i} = E[x(n-k)x(n-i)]$$

and  $\mathbf{p}$  is the cross correlation vector between the input and desired signals defined as  $\mathbf{p} = [p_0 \dots p_{M-1}]^T$ ,  $p_k = E[z(n)x(n-k)]$  [141].

Instead of analytically computing the optimal solution with Eq. (1.30), one can use a search technique such as steepest descent (also known as gradient descent). Indeed, Eq. (1.26) defines a cost function in the space of the weights which is conventionally called the *performance surface* [332]. Due to the linear and feedforward nature of the FIR, the performance surface contains a single minimum. The gradient of the error is obtained as

$$\nabla \mathbf{J}(n) = \frac{\partial J(e(n))}{\partial \mathbf{w}} = \frac{\partial E(e^2(n))}{\partial \mathbf{w}}. \quad (1.31)$$

An initial weight vector is chosen (normally  $\mathbf{w}(0) = 0$ ), the gradient at the point is estimated, and a new weight vector is found by changing  $\mathbf{w}(n)$  proportionally to the negative of the gradient vector; that is,

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \eta \nabla \mathbf{J}(n). \quad (1.32)$$

The procedure is repeated until the operating point is in the neighborhood of the optimal solution found by analytical methods. The constant of proportionality  $\eta$  is called the *stepsize* or the learning rate. This expression can be applied locally to each of the weights in the weight vector. If the expected value is substituted by the average over  $N$  past samples, the gradient can be estimated from the available data at sample  $n$ . This steepest descent algorithm has been studied in depth in the adaptive filtering literature [332] and it is known that the stepsize for convergence is upper bounded by the inverse of the largest eigenvalue of the input data autocorrelation matrix  $\mathbf{R}$ ;  $\eta < 1/\lambda_{\max}$ . The famed least mean square algorithm is a stochastic approximation to the steepest descent that uses the current error ( $e(n)$ ) as the gradient estimate; that is,

$$\begin{aligned} \nabla_k \mathbf{J}(n) &= \frac{\partial}{\partial w_k} \left( \frac{1}{2N} \sum_{i=0}^{N-1} e^2(n-i) \right) \approx \frac{\partial}{\partial w_k} (e^2(n)) = \frac{\partial e^2(n)}{\partial e(n)} \frac{\partial e(n)}{\partial w_k} \\ &= -e(n)x_k(n) \end{aligned} \quad (1.33)$$

yielding the simple LMS adaptation formula

$$w_k(n+1) = w_k(n) + \eta e(n)x_k(n), \quad k = 0, \dots, M-1 \quad (1.34)$$

which has a complexity of two multiplications per weight. The computation in Eq. (1.34) is also local in the topology and local in time. The simplicity of this algorithm launched the engineering applications of optimal filtering as we know them today.

The least square solution which uses the MSE cost has a very distinguished history dating back to Gauss, and leads to a closed-form characterization of the solution as a conditional mean. But it can also be framed in statistical terms as regression. Given two dependent random variables  $\{x(n), z(n)\}$ , we can pose the problem of determining an “optimal” estimator of  $z(n)$  given  $x(n)$ . Under the minimum mean-square-error criterion (MMSE), we have

$$\min_w E[e^2(n)] \quad (1.35)$$

with  $e(n) = z(n) - f(x(n), \mathbf{w})$ , and  $E[e(n)] = 0$ . It is well known that  $\hat{z}(n)$ , the minimum least mean square estimator of  $z(n)$  given  $x(n)$ , is the conditional expectation of  $z(n)$  given  $x(n)$  [235]:

$$y(n) = E[z(n) | x(n)] = \int z(n)p(z(n) | x(n))dz, \quad (1.36)$$

which corresponds to the minimization of the error variance of Eq. (1.35). Moreover,

$$\begin{aligned} E[\hat{z}(n)] &= \bar{y}(n) \\ E[e^2(n)] &= E[z^2(n)] - E[y^2(n)] \rightarrow \sigma_z^2 \geq \sigma_y^2 \\ E[e(n)x(n)] &= 0. \end{aligned} \quad (1.37)$$

There are several important observations. First, the result  $\sigma_z^2 \geq \sigma_y^2$  is interesting because, if we view the variance as a loose measure of uncertainty (similar to entropy), this inequality tells us that the “information” contained in the estimate cannot be more than that of the original desired random variable.

Second, the orthogonality condition in Eq. (1.37) states that an estimator  $y(n) = \hat{z}(n) = f(x(n))$  is optimal in the least mean squares sense if and only if  $y(n)$  is unbiased and  $e(n) \perp f(x(n))$  for any function  $f$ . In other words, the optimal MMSE estimator is such that no matter how we modify the system parameters, there is no better solution that can extract additional structure to reduce the *variance of  $y(n)$*  further. In other words the orthogonality condition is a defining property for the minimization of the variance. Effectively this corresponds to estimating the orthogonal projection (with an Euclidean norm) of the desired response  $z(n)$  in the space spanned by the states of the system, or for FIR filters, the space spanned by the input signal  $x(n)$ .

However, a more general question is under what conditions is the choice of MSE optimal because the optimization only constrains the second-order moment (variance) of the error, this question is equivalent to optimizing the second-order statistic. For instance, Wiener [333] has shown that if the signals  $x(n)$  and  $z(n)$  are wide sense stationary random processes (i.e., fully described by second-order statistics) then this procedure is optimal. In a statistical framework, as long as the residuals are Gaussian, least squares will provide the best linear fit.

## 1.8 Information-Theoretic Learning

When the residuals are not Gaussian distributed, a more appropriate approach would be to constrain directly the information content of signals rather than simply their energy [68, 177, 199]. Because entropy is defined as the uncertainty of a random variable, it is only natural to adopt it as the criterion for applications where manipulation of the information content of signals is desired or necessary. However, there are important differences between the application of information theory to communication systems and the reality of adaptive signal processing and machine learning.

1. Adaptive systems must handle continuous-valued random processes rather than discrete-valued processes. Noting this fact, we must focus our discussion on continuous random variables, described by their probability density functions.
2. Adaptive algorithms require smooth cost functions; otherwise the local search algorithms become difficult to apply.
3. The data statistics in machine learning and modern signal processing applications have long tails (especially when nonlinear topologies are considered) and the real-world examples are plagued with outliers. Therefore, the Gaussian assumption so widespread in communications is normally a poor descriptor for these applications.

This means that the analytic approach taken in information theory must be modified with continuous and differentiable nonparametric estimators of entropy and divergence. To meet requirements 2 and 3 we are convinced that the nonparametric kernel density estimators championed by Rosenblatt and Parzen [241, 272] are a productive research direction. As we show, kernel (Parzen) estimation has the added advantage of linking information theory, adaptation, and kernel methods.

This book provides a general ITL methodology to implement adaptive algorithms with information theoretic criteria. ITL synergistically integrates the general framework of information theory in the design of new cost functions for adaptive systems, and it is poised to play an expanding role in adaptive signal processing. ITL does not only affect our understanding of optimal signal processing, but also influences the way we approach machine learning, data compression, and adaptation as we demonstrate in the sequel.

The fundamental issue in ITL is how to estimate entropy and divergence directly from samples, which is treated in Chapter 2 in great detail. Chapters 3 to 5 cover the learning algorithms to adapt linear or nonlinear systems using the error entropy criterion (EEC). ITL also provides divergence and mutual information measures and the corresponding estimators. These mesoscopic descriptors can also be used to train linear and nonlinear systems when a desired response is available as in classification (Chapter 6), or even extract structure from the input data alone (unsupervised learning) as demonstrated in Chapters 7 and 8.

## 1.9 ITL as a Unifying Learning Paradigm

The theory of learning is conventionally divided in three basic principles: supervised, unsupervised, and reinforcement learning. Although there are good reasons to make this division, the algorithms that are currently employed accentuate the differences. We will briefly present the impact of ITL for learning theory in this section.

### Supervised Learning Paradigm

In supervised learning, the adaptive system has access to two sources of data from the external world; pairs of signals  $\{x_i, z_i\}$ . The goal of learning is to discover the model that relates  $x$  with  $z$  (Figure 1-1). Given the existence of two external data sources, the supervised learning problem normally is framed as a functional mapping, where there is a parametric system in some functional class that receives the input  $x$  and  $z$  takes the role of the desired response. Hence, an error between the desired and the adaptive system output  $y$  can be easily defined by subtraction, which leads to the idea of penalizing the error for adaptation and to find the parameters that achieve this minimal error. This gives rise to least squares and gradient descent procedures, which became the hallmarks of supervised adaptation.

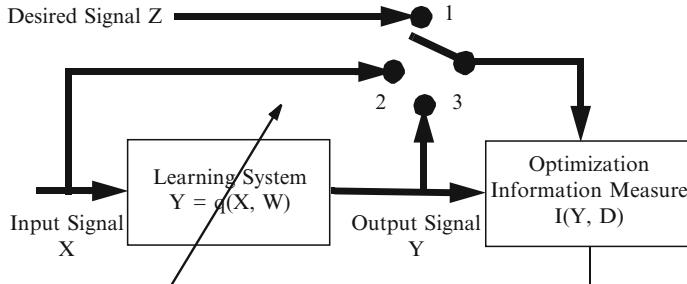
### Unsupervised Learning Paradigm

In unsupervised learning, the adaptive system has access to a single source of data from the external world. The goal of learning in this case is to find system parameters that quantify the data structure. In principle it is not easy to pose the unsupervised learning problem in terms of a cost, but one of the exciting advances of the last 20 years has been the formulation of self-organizing principles [152] that are also related to “energy methods” [196].

### Reinforcement Learning Paradigm

In reinforcement learning, there are still two sources of data from the external world, however, one of them is just a binary value that tells the system if its response is good or bad. Therefore, the supervised learning methodologies cannot be directly applied, and the system has to explore the environment to receive rewards. Although supervised learning has thus far been the major inspiration for reinforcement learning algorithms, there is a role for unsupervised methods to help structure the search. This topic is not treated further here.

The ITL methodologies play an important role to develop algorithms that are independent of the learning paradigm; that is, the same algorithm can be applied to both unsupervised and supervised scenarios just by switching the inputs to the cost function. This characteristic unifies the learning paradigms and is shown in Figure 1.9. ITL accomplishes this goal by creating versatile cost functions that are external to the learning machine and also by framing unsupervised algorithms as adaptations with an external cost function.



**Fig. 1.9.** A unifying view of learning based on ITL cost functions (from [252]).

### Divergence or Mutual Information Criteria

One of the inputs to the cost function in Figure 1.9 is always the system output. The control of supervised versus unsupervised is done by the switch that brings the other source of data to the cost criterion. Instead of mutual information as shown in the figure, divergence can be used interchangeably, except for the fact that minimization will be substituted by maximization of the cost.

#### Switch in position 1

*Filtering (or regression) and classification:* When the switch is in position 1, the cost receives the system output and the other source of information is the desired response, so the problem is supervised in our terminology (two sources of data from the external world), and by minimizing the divergence (or maximizing MI) we are making the system output as close as possible to the desired response, just as in regression or optimal filtering with the minimization of the MSE, except that now we are using knowledge about the PDFs, and we do not require the same number of system outputs as desired responses, as demonstrated in Chapter 6.

$$\begin{aligned} \min_{\mathbf{w}} D_{KL}(\mathbf{y} \mid z) &= \int p(\mathbf{y}) \log \frac{p(\mathbf{y})}{p(z)} d\mathbf{y} \\ \max_{\mathbf{w}} I(\mathbf{y}, z) &= H(\mathbf{y}) - H(\mathbf{y} \mid z). \end{aligned} \quad (1.38)$$

The same thing happens for classification, except that now the desired response is a set of indicator functions (discrete value function; i.e., 0/1). This might simplify the calculations.

*Feature Extraction:* Suppose that the desired response is an indicator function. In this case, one important question is how to project the high-dimensional input to a possibly nonlinear subspace, such that the discriminability of the outputs with respect to the labels is preserved. The problem can be solved by

maximizing the mutual information between the projection outputs and the class labels:

$$\max_{\mathbf{w}} I(\mathbf{y}, c) = H(\mathbf{y}) - \sum_c p_c H(\mathbf{y} \mid c). \quad (1.39)$$

This principle generalizes the concepts behind PCA and LDA to nonlinear feature projections [106, 108].

### Switch in position 2

When the switch is in position 2, the other source is the input data itself, so learning is unsupervised according to our definition because the cost has access to the data ( $x$ ) and a processed version of the data ( $y$ ).

*Maximum Information Transfer:* The optimization problem is to maximize the transfer of information between the input and the output of the system. This is called the principle of maximal information transfer and is related to the channel capacity theorem and the information bottleneck framework [97, 316]. One could maximize the mutual information between the original input data and the transformed output data to preserve information maximally while reducing noise.

$$\max_{\mathbf{w}} I(\mathbf{y}, \mathbf{x}) = \int p(\mathbf{y}, \mathbf{x}) \log \frac{p(\mathbf{y}, \mathbf{x})}{\prod_d p(\mathbf{y}_d)p(\mathbf{x}_d)} d\mathbf{y} d\mathbf{x}. \quad (1.40)$$

This formulation has also been suggested as a self-organization principle in distributed systems.

### Switch in position 3

When the switch is in position 3, the only source of data is the system output that is assumed multidimensional.

*Independent Component Analysis:* In this case if we minimize MI we are performing redundancy reduction or independent component analysis, which is an unsupervised problem. Indeed, one assumes a multiple-input–multiple-output (MIMO) system and the goal is to create statistically independent outputs [57, 156]. For a nonlinear MIMO system  $\mathbf{y} = \mathbf{f}(\mathbf{x}; \mathbf{w})$ , the nonlinear ICA problem seeks to determine the parameters  $\mathbf{w}$  of  $\mathbf{f}(\cdot)$  such that the mutual information between the components of  $\mathbf{y}$  is minimized (preferably to zero):

$$\min_{\mathbf{w}} I(\mathbf{y}, y_1 \dots y_d) = \int p(\mathbf{y}, y_1 \dots y_d) \log \frac{p(\mathbf{y}, y_1 \dots y_d)}{\prod_d p(y_d)} d\mathbf{y}. \quad (1.41)$$

*Clustering:* Finally, assume that the goal of the mapper is to divide the input data into a preselected number of structurally and/or statistically distinct groups (clusters). Here, the weights become the assigned cluster membership values and the criterion is to assign samples to a cluster such that the clusters are defined as compactly and distinctly as possible, measured by cluster

entropy and divergence. In the case of two clusters, one could use the J divergence, for example;

$$\max_{\mathbf{w}} D_{KL}(p_1(\mathbf{y}) \parallel p_2(\mathbf{y})) + D_{KL}(p_2(\mathbf{y}) \parallel p_1(\mathbf{y})). \quad (1.42)$$

## Entropy Costs

Instead of using divergences or mutual information, we can also use the entropy as our information cost, but now only one variable is utilized for the cost, so an extra operation may be necessary before bringing the signal to the cost.

### Switch in position 1

*Error Entropy Criterion:* With the switch in position 1, if the difference between the desired and the system output is computed first (i.e., the error) and the criterion minimizes entropy, this is information filtering (or information regression) and classification.

$$\min_{\mathbf{w}} H(e) = \int p(e) \log p(e) de. \quad (1.43)$$

### Switch in position 2

*Optimizing for Extremes of System Output:* An alternative is to simply maximize (or minimize) the entropy at the system output (subject to some constraint on the weight vector norm or the nonlinear topology), which leads to an information-theoretic factor analysis to discover interesting structures in the high-dimensional input data.

$$\max_{\mathbf{w}} H(\mathbf{y}) = - \int p(\mathbf{y}) \log p(\mathbf{y}) d\mathbf{y} \quad (1.44)$$

$$\text{subject to } E[h_i(\mathbf{y})] = \alpha_i, \quad i = 1, \dots, d.$$

This formulation is useful in blind equalization, nonlinear principal component analysis, ICA, and novelty filtering [139, 156].

On the other hand, if the switch is in position 3 and the entropy criterion is maximized, we are implementing MaxEnt. All these cases are summarized in Table 1.1-1.3. Perhaps even more remarkably as we show in Chapter 2, all of these learning scenarios can be implemented using nonparametric estimators of Renyi's entropy and Euclidean distance or Cauchy–Schwarz divergence based on the *information potential* (for entropy) or the *generalized information potential* (for divergence or mutual information).

Therefore the ITL methodology improves upon many of the previous works where the self organizing principles were applied but their implementation with real data used Gaussian assumptions, wasting the generality of the principles in the sense that equivalent maximum likelihood solutions could be used instead.

**Table 1.3.** Summary of ITL costs for Different Applications

Switch	Distance Max	Distance Min	Mutual Info Max	Mutual Info Min	Entropy Max	Entropy Min
1		Information filtering	Information filtering	Innovation filtering		Information filtering
2		InfoMax	InfoMax			
3	Clustering			ICA	MaxEnt (NLPCA)	

## 1.10 Reproducing Kernel Hilbert Spaces

A Hilbert space is a linear, complete, and normed space endowed with an inner product [214]. Some Hilbert spaces are infinite-dimensional vector spaces and they are the foundation of continuous-time signal processing. Prior to the widespread use of digital signal processing, which is based on finite discrete time series of finite length that can be properly described in Euclidean spaces ( $R^d$ ), infinite-dimensional Hilbert spaces were commonly used. Fourier transforms, Karhunen–Loeve transforms, and Wiener filters were initially studied in infinite-dimension Hilbert spaces, so the theory is very well developed. The existence of a reproducing kernel Hilbert space corresponding to any symmetric and nonnegative definite kernel function is one of the most fundamental results of the theory. The RKHS framework was originally studied because all Green's functions of self-adjoint ordinary differential equations and some bounded Green's functions in partial differential equations belong to this special class of functionals of two variables. But it was not until 1943 that Aronszajn systematically developed the general theory of RKHS and coined the term “reproducing kernel” [7]. The expanded paper [8] on his previous work became one of the standard references for RKHS theory.

The application of RKHS methodology in statistical signal processing was proposed by Parzen in the late 1950s, who provided for the first time a functional analysis perspective of random processes defined by second-order moments (called Gaussian processes) because they can be approached by purely geometric methods when studied in terms of their second-order moments (covariance kernel) [238]. Parzen clearly illustrated that the RKHS approach offers an elegant general framework for minimum variance unbiased estimation of regression coefficients, least squares estimation of random variables, detection of known signals in Gaussian noise, and so on. Although they involve random variables, all these problems can be solved algebraically in the RKHS associated with their covariance functions with all the geometric advantages of the inner product defined in such spaces. In the early 1970s, Kailath presented a series of detailed papers on the RKHS approach for detection and estimation to demonstrate its superiority in computing likelihood ratios, testing for non-singularity, bounding signal detectability, and determining detection stability [173–175]. Although the approach was very elegant, it did not produce new

results. RKHS concepts have also been extensively applied to a wide variety of problems in optimal approximation by Wahba including minimum norm interpolation and smoothing by spline functions in one or more dimensions (curve and surface fitting) [327]. Figueiredo took a different approach to apply RKHS in nonlinear system and signal analysis [69]. He built the RKHS bottom-up using arbitrarily weighted Fock spaces. The spaces are composed of Hilbert–Schmidt polynomials or power series in either scalar or multidimensional variables. The generalized Fock spaces have been applied in engineering for nonlinear system approximation, semiconductor device characteristics modeling, and neural networks [103].

Today, RKHS are often applied in connection with the kernel methods of machine learning [289, 294], and it is important to understand their advantage for this application, which is different from the work developed by Parzen. Since Cover’s work [232] we know that the probability of linearly “shattering” data (i.e., finding a hyperplane that classifies the data with zero error) approaches one with the increase in dimensionality of the space. However, the bottleneck of this technique was the large number of free parameters of the high-dimensional classifiers, hence computation would become expensive and there would be the need to regularize the solutions. The RKHS provides a way to simplify the computation, because for most kernels the space dimension is very high (even infinite), but by the “*kernel trick*” the calculations can still be done efficiently in the input space provided the quantities of interest in the algorithms can be expressed by inner products. However, the problem of regularization still remains. The work on support vector machines by Vapnik rekindled the interest in RKHS for pattern recognition [232] because he provided a robust regularizer which is essential when working in high-dimensional spaces with analytic solutions. The excitement of the kernel methods is very vivid today. Essentially a kernel machine is a one-layer neural network (e.g., RBF when Gaussian kernels are used) whose parameters can be analytically computed in the RKHS given the training set data using either the structural risk minimization principle [323] or least square techniques [311].

A linear system in RKHS may become (depending upon the kernel) a nonlinear filter in the input space, and this opens a very interesting avenue to pursue nonlinear signal-processing applications. If an optimal iterative solution is desirable, gradient descent algorithms have also been proposed in RKHS [202], with the advantage that there are no local minima in the adaptation. This is an enormous advantage with respect to neural networks, where the training is always problematic because the system state can be caught in local minima. Therefore developing adaptive solutions in RKHS is an intriguing possibility to go beyond traditional neural networks.

RKHS also appear naturally in information-theoretic learning. In fact, inasmuch as we are interested in nonparametric approaches to estimate entropy and divergence, kernel density estimation is central in ITL. There is a large overlap between the mathematical conditions required for a kernel proper for density estimation and positive definite functions (in fact most of

the kernels used in density estimation are positive definite). However, there are even deeper connections as Figure 1.2 illustrates through the cross-information potential. It turns out that the argument of quadratic Renyi's cross-entropy is a positive definite function involving PDFs. Therefore, it is possible to define a RKHS for ITL as explained in Chapter 9. We now briefly review RKHS foundations.

## RKHS Definitions

A reproducing kernel Hilbert space is a special Hilbert space associated with a kernel  $\kappa$  such that it reproduces (via an inner product) each function  $f$  in the space, or, equivalently, every point evaluation functional is bounded [214]. Let  $H_\kappa$  be a Hilbert space of real-valued functions defined on a set  $E$ , equipped with an inner product  $\langle \cdot, \cdot \rangle$  and a real-valued bivariate function  $\kappa(x, y)$  on  $E \times E$ . Then the function  $\kappa(x, y)$  is said to be nonnegative definite if for any finite point set  $\{x_1, x_2, \dots, x_n\} \subset E$  and for any not all zero corresponding real numbers  $\{\alpha_1, \alpha_2, \dots, \alpha_n\} \subset R$ ,

$$\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j \kappa(x_i, x_j) \geq 0. \quad (1.45)$$

Any nonnegative definite bivariate function  $\kappa(x, y)$  is a reproducing kernel because of the following fundamental theorem.

**Theorem 1.1 (Moore–Aronszajn).** *Given any nonnegative definite function  $\kappa(x, y)$ , there exists a uniquely determined (possibly infinite-dimensional) Hilbert space  $H_\kappa$  consisting of functions on  $E$  such that*

- (I)  $\forall x \in E, \kappa(\cdot, x) \in H,$
- (II)  $\forall x \in E, \forall f \in H, f(x) = \langle f, \kappa(\cdot, x) \rangle_{H_\kappa}.$

By property (I) we see that each point in the input space is mapped onto a function in the RKHS defined by the selected kernel. Therefore the richness of the representation in RKHS is related to the kernel one defines, thus we denote this dependence by  $H_\kappa$ , or  $H := H_\kappa$  is said to be a reproducing kernel Hilbert space with reproducing kernel  $\kappa$ . The property (II) is called the *reproducing property* of  $\kappa(x, y)$  in  $H_\kappa$ . In particular, we can define our nonlinear mapping from the input space to a RKHS as  $\Phi(x) = \kappa(\cdot, x)$ ; then we have

$$\langle \Phi(x), \Phi(y) \rangle_{H_\kappa} = \langle \kappa(\cdot, x), \kappa(\cdot, y) \rangle = \kappa(x, y) \quad (1.47)$$

and thus  $\Phi(x) = \kappa(\cdot, x)$  defines the Hilbert space associated with the kernel. Notice that similarity between functions in the RKHS is also totally defined by the kernel because it defines the inner product of functions. For those who know delta functions from signal processing, the delta function has the reproducing property (through convolution it extracts the value of the function at

the point where it occurs), but does not obey Property I above, so it is not a kernel nor defines a RKHS. Besides its defining property, this theorem also suggests a huge simplification which is addressed next.

## Kernel-Based Learning Theory

Kernel-based (Mercer) learning algorithms use the following idea [289]. Via a nonlinear mapping

$$\begin{aligned}\Phi : E &\rightarrow H \\ x &\rightarrow \Phi(x)\end{aligned}$$

the data  $\{x_1, x_2, \dots, x_n\} \subset E$  (where  $E$  is usually  $R^d$ ) are mapped into a potentially much higher-dimensional feature space  $H_\kappa$  with a linear structure. A given learning problem in  $E$  is solved in  $H_\kappa$  instead, by working with  $\{\Phi(x_1), \dots, \Phi(x_n)\} \subset H$ . Because  $H_\kappa$  is high-dimensional, a simple linear learning algorithm, and preferably one expressed solely in terms of inner-product evaluations, can solve arbitrarily nonlinear problems in the input space (if  $H_\kappa$  is sufficiently rich to represent the mapping). The inner product formulation implicitly executes the linear algorithm in kernel feature space but the data and the operations are all done in the input space (by the kernel property of Eq. (1.47), normally called the kernel trick). The Mercer theorem guarantees the existence of the nonlinear mapping  $\Phi$ .

**Theorem 1.2 (Mercer's).** *Consider a symmetric kernel function  $\kappa \in L_\infty(E \times E)$ . If  $\kappa$  is the kernel of a positive integral operator in  $L_2(E)$ , and  $E$  is a compact subset of  $R^d$  then*

$$\forall \psi \in L_2(E) : \int_E \kappa(x, y) \psi(x) \psi(y) dx dy \geq 0. \quad (1.48)$$

Let  $\Phi_i \in L_2(E)$  be orthonormal eigenfunctions of the above operator and  $\lambda_i > 0$  their corresponding eigenvalues. Then

$$\kappa(x, y) = \sum_{i=1}^{N_F} \lambda_i \Phi_i(x) \Phi_i(y) \quad (1.49)$$

holds for  $N_F < \infty$  or  $N_F = \infty$ . In the latter case the series converges absolutely and uniformly for almost all  $x$  and  $y$  in  $E$  [217]

The operation in Eq. (1.49) clearly provides a nonlinear mapping via the eigenfunctions determined by the kernel. The kernel trick can be used to develop nonlinear generalizations of any algorithm that can be cast in terms of inner products [287]. For example, KPCA, KLDA, and kernel k-means [72, 112, 287] are simply extensions of the corresponding linear algorithms by applying the kernel trick on every inner-product evaluation. A kernel that satisfies Eq. (1.49) is known as a Mercer kernel. The most widely used Mercer

kernel is the Gaussian function (also used in the radial basis function (RBF) network)

$$G_\sigma(x, y) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right) \quad (1.50)$$

where  $\sigma$  is a kernel size or bandwidth that works as a scale parameter which controls the width of the Gaussian. A Gaussian kernel corresponds to an infinite-dimensional Mercer kernel feature space, because the Gaussian has an infinite number of eigenfunctions.

Nevertheless, there is an immediate downside to this property, and it appears as the need to evaluate the functions in a pointwise manner; that is the algorithms become memory intensive because in principle all the kernel evaluations must be saved to compute future outputs. In fact, these learning machines look like nonparametric algorithms when implemented with the kernel trick, except that there is a model in the RKHS to adapt the parameters. After this brief presentation of RKHS properties, RKHS is linked with information theory, adaptive filtering which is an example of the RKHS for representation, and their role for statistical inference.

## 1.11 RKHS and ITL

The need to estimate nonparametrically entropy and divergence raises the question of how to accomplish this goal. Because these descriptors are based on the PDF, the use of kernel density estimation jumps immediately into our minds because of its nice properties. Kernel density estimation is a very well established field [272], therefore many results are available quantifying the bias and variance of the estimators and their consistency [241], as well as its difficulties in determining the optimal kernel size and its poor efficiency in high-dimensional spaces [170].

Most of the kernels used in density estimation are indeed nonnegative bivariate functions, therefore they define a RKHS. For instance, Schölkopf shows that classifiers can be easily constructed using the mean of the transformed data, which implicitly use the PDF as the discriminant function [225]. However, there are deeper connections that are explored in this book as illustrated in Figure 1.2. Let us define cross-entropy between two PDFs  $p(x)$  and  $q(x)$  as

$$H(p; q) = - \int p(x) \log q(x) dx = -\mathbb{E}_p[\log q(x)], \quad (1.51)$$

which measures the average number of bits needed to identify an event from a density  $q(x)$  with a coding scheme built on a different probability density  $p(x)$ . For Renyi's entropy, the equivalent quadratic cross-entropy is defined as

$$H_2(p; q) = -\log \int p(x)q(x) dx = -\log \mathbb{E}_p[q(x)]. \quad (1.52)$$

It turns out that the argument of the logarithm, which we call the cross information potential, is a positive definite function as we demonstrate in Chapter 9 so it defines a RKHS that will provide a functional analysis view of the information-theoretic descriptors of entropy and divergence. Note that for the special case  $p(x) = q(x)$ , the cross information potential defaults to the information potential which is the argument of the 2-norm of the PDF  $p(x)$ . The IP appears in a prominent role in Figure 1.2 because it is used in Renyi's quadratic entropy, and it is also the cost function that is used to adapt linear or nonlinear systems in Chapters 3, through 6. The CIP is utilized as the basis for similarity in clustering and unsupervised learning applications in Chapters 7 and 8.

Let us contrast the RKHS defined by the CIP with the RKHS defined in kernel methods. Let  $E$  be the space of input samples  $\{x_i\}_{i=1}^N$ . A kernel is an inner-product operator  $\kappa : E \times E \rightarrow R$ . An explicit way to describe  $\kappa$  is via a mapping  $\Phi : E \rightarrow H$  from  $E$  to an inner-product space  $H_\kappa$  such that  $\kappa(x, x') = \langle \Phi(x) \cdot \Phi(x') \rangle$ . In this RKHS the user selects the kernel as any positive definite function (Gaussian, Laplacian, etc.), therefore the kernel function is independent of the data, and if the input space data are random variables, the RKHS is built from functions that operate on stochastic data producing stochastic outputs. The dimensionality of the RKHS is controlled by the dimension of the mapping  $\Phi$ , which is potentially very large (hence its primary use for representation). The size of the kernel matrix (Gram matrix) is  $N \times N$ , its elements are random variables, and statistical operators are needed to operate on the Gram matrix.

The RKHS defined by the argument of the logarithm in the CIP, Eq. (1.52) is vastly different. The input space  $E$  is now the set of all one-dimensional and square integrable PDFs defined in the sample space; that is,  $f_i(x) \in E$ ,  $\forall i \in I$ . The kernel is still an inner product operator defined on  $\kappa : E \times E \rightarrow R$  but the mapping between  $E$  and  $H_v$  is now defined by the CIP (i.e.  $\mathcal{V}(f_i, f_j) = \int f_i(x) f_j(x) dx$ ,  $\forall i, j \in I$ ). As one can see, the kernel now is built explicitly with the statistical properties of the input data. The elements of  $H_v$  are deterministic functions so one can operate algebraically with them for statistical inference. The dimensionality of this RKHS is controlled by the dimension of the PDF function, and the size of the data matrix is the size of  $I$ . There is a relationship between the two RKHS that is further elaborated in Chapter 9. We now present two simple applications of the use of these two classes of RKHS.

## An Adaptive Filter in RKHS

The purpose of this example is to illustrate the use of RKHS to design optimal nonlinear adaptive filters to bring together RKHS and adaptive filter theory. The RKHS is defined on the sample set of the input and the inner product is defined by the kernel selected for the transformation, therefore it is an example of the RKHS for representation. As the simplest of the examples, we

derive the equations to train in RKHS a linear adaptive FIR filter with the LMS algorithm. It is well known that the optimal FIR filter  $\mathbf{w}$  between r.v.  $X$  and  $Z$  (Figure 1.1) is defined as

$$y(n) = \mathbf{w}^T \mathbf{x}(n) = \langle \mathbf{w}, \mathbf{x}(n) \rangle, \quad (1.53)$$

where the size of the vector of weights and inputs is established by the filter order  $M$ . It is also well known that the optimal weight vector is given by  $\mathbf{w}^* = \mathbf{R}^{-1} \mathbf{p}$  with the definitions in Eq. (1.25). Notice from Eq. (1.53) that the FIR filter can be written as an inner product between the weight vector and the input, so optimal filtering is a good candidate for a RKHS implementation and the end result will be a nonlinear optimal filter in the input space. The equation analogous to Eq. (1.53) for a linear filter in RKHS is therefore

$$y(n) = \langle \Omega(n), \Phi(\mathbf{x}(n)) \rangle_{H_\kappa}, \quad (1.54)$$

where  $\Phi(\mathbf{x}(n))$  is the transformed data  $\mathbf{x}(n)$ , and  $\Omega(n)$  is the weight function (the filter) in the RKHS  $H_\kappa$  at iteration  $n$ , and the inner product is defined by the selected kernel (e.g., the Gaussian kernel). So conceptually, what is needed is to map the input by the nonlinear function specified by the kernel eigenfunctions, and do an inner product with the weight vector, which can be computed in the input space by the kernel evaluation.

The fundamental issue is how to find the optimal weights, and here instead of attempting to solve the least square problem analytically as done in kernel methods [289], we proceed with an implementation of the kernel LMS (KLMS) algorithm [202] for simplicity. If you recall, the LMS algorithm is given by

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \eta e(n) \mathbf{x}(n), \quad (1.55)$$

where  $\eta$  is the stepsize that needs to be selected by the user. Let us now implement this recursion in kernel spaces. Let us start with a zero weight function  $\Omega(0) = 0$ , which we can think of as a high-dimensional vector. The error at iteration  $n$  is given by

$$e(n) = d(n) - \langle \Omega(n-1), \Phi(\mathbf{x}(n)) \rangle_{H_\kappa} \quad (1.56)$$

inasmuch as it is the output of the system computed with the previous weights. Now the new weight vector from Eq. (1.55) becomes

$$\Omega(n) = \Omega(n-1) + \eta \Phi(\mathbf{x}(n)) e(n). \quad (1.57)$$

It is easy to show that we can write  $\Omega(n-1)$  in terms of  $\Omega(n-2)$  until  $\Omega(0)$  that can be set at zero and rewrite Eq. (1.57) iteratively from the first weight to obtain

$$\Omega(n) = \sum_{j=1}^n \eta e(j) \Phi(\mathbf{x}(j)). \quad (1.58)$$

Therefore, we can now compute the present output substituting Eq. (1.58) into Eq. (1.54)

$$\begin{aligned} y(n) &= \langle \Omega(n), \Phi(\mathbf{x}(n)) \rangle_{H_\kappa} = \left\langle \sum_{j=1}^n \eta e(j) \Phi(\mathbf{x}(j)), \Phi(\mathbf{x}(n)) \right\rangle_{H_\kappa} \\ &= \sum_{j=1}^n \eta e(j) \kappa(\mathbf{x}(j), \mathbf{x}(n)). \end{aligned} \quad (1.59)$$

This iteration if it converges will approximate the optimal least square solution in the RKHS and will provide a nonlinear filter in the input space. For further analysis of this very interesting solution that does not need explicit regularization, see [202].

Let us now discuss Eq. (1.59). First, if the kernel is a Gaussian, the nonlinear filter is a radial basis function (RBF) network with a growing structure, where centers are placed at the projected samples and the weights are the errors at each sample. The filter grows with each new sample following a non-parametric approach that is discovering the mapping as time progresses. Of course, because of finite resources one must cut the growing structure of the filter, but if one recalls the learning curve in LMS the error decreases exponentially with the number of samples until it stabilizes. A similar behavior is displayed by the KLMS, so a relatively small number of centers provide better solutions than the LMS in nonlinear problems [202]. It is also possible to compute in RKHS the optimal regressor using a block of data, but the solution needs regularization and is essentially equivalent to the kernel regression solution (see [225]). We do not pursue these developments here, but it is important to keep in mind that the ITL cost functions developed throughout this book can also be used to train filters in RKHS.

## RKHS Induced by the Covariance of a Random Process

The theory of Hilbert space representations of Gaussian processes (i.e., random functions or stochastic processes  $\{X(t), t \in T\}$  fully described by second-order statistics) was mentioned by Loeve [203] but mostly developed by Parzen [238]. Our explanation will follow [238] very closely. This RKHS is defined in an index set  $T$  and it is induced by the covariance function of the random process, which is different in nature from the Gaussian kernel used in the previous section. Therefore, we briefly study here some of its properties.

A time series is a family  $X(.) = \{X(t), t \in T\}$  of random variables with finite second order moments. The mean value function  $m$  and the covariance kernel  $R$  are functions on  $T$  and  $T \times T$  defined as  $m(t) = E[X(t)]$  and  $R(s, t) = \text{Cov}[X(s), X(t)]$  respectively.

It is often convenient to think of a time series  $X(.)$  as indexed by a Hilbert space  $H_o$  with inner product  $\langle x, y \rangle_{H_o}$  in the sense that the “random” inner product  $\langle X, h \rangle_{H_o}$ , where  $h$  is a vector in  $H_o$  is not a true inner product

but represents a random variable indexed by  $h$ . The basic goal to define this random inner product is to establish a correspondence between elements of  $H_o$  and random variables.

Let us define the Hilbert space  $H_X$  spanned by a Gaussian process  $\{X(t), t \in T\}$  to be the set of all random variables in the linear manifold  $L(X(t), t \in T)$  together with their limit points and denote it by  $L_2(X(t), t \in T)$ , i.e.  $H_X = H(X(t), t \in T) = L_2(X(t), t \in T)$ . The Hilbert space spanned by the time series becomes the smallest Hilbert space of random variables that contains  $X(\cdot)$ . It can be shown that every random variable  $U$  in  $H_X$  is a linear functional of  $X(\cdot)$ , i.e.  $U = \sum_{j=1}^n c_j X(t_j)$ , for some  $n$ . Parzen proved the following theorem:

**Theorem 1.3 (Parzen).** *For a stochastic process  $\{X(t), t \in T\}$  with  $T$  being an index set,  $R(s, t)$  is the covariance function, if and only if, it is symmetric and positive definite.*

An immediate result from Theorem 1.3 and 1.1 is that  $R$  also defines a RKHS  $H_R$  for the stochastic process. In the proof of Theorem 1.3, Parzen also showed that for any symmetric and positive definite kernel function there is a space of Gaussian distributed random variables defined on the same domain for which the kernel is the covariance function. In other words, any kernel inducing a RKHS denotes simultaneously an inner product in the RKHS and a covariance function in the space of random processes. Furthermore, Parzen established that there exists an isometric isomorphism (i.e., a one-to-one inner product-preserving mapping), also called a congruence, between these two spaces. This is an important result as it sets up a correspondence between the inner product due to the kernel in the RKHS to our intuitive understanding of the covariance function and associated statistics. Let us explore these ideas more formally.

Let us start by obtaining a representation of a random variable which is a function of a stochastic process  $\{\theta(s), s \in S\}$ . Parzen showed that [238]:

1. Every function  $f$  in  $H_R$  can be represented has  $f(s) = E[\theta(s)U]$  for some unique  $U$  in  $L_2(\theta(s), s \in S)$  with zero mean and variance  $\|f\|_{H_R}^2$ .
2. A one to one correspondence between  $H_R$  and  $L_2(\theta(s), s \in S)$  exists such that

$$\begin{aligned}\theta(s) &\leftrightarrow R(., s) \\ U = \sum_i c_i \theta(s_i) &\leftrightarrow f = \sum_i c_i R(., s_i)\end{aligned}$$

The function  $f \in H_R$  that corresponds to the random variable  $U \in L_2(\theta(s), s \in S)$  is denoted by  $\tilde{f}$  or  $\langle f, \theta \rangle_{\tilde{H}_R}$  and will be called a congruence inner product. With this result one can write

$$\begin{aligned}E \left[ \langle f, \theta \rangle_{\tilde{H}_R} \right] &= \langle f, \theta \rangle_{H_R} \\ Cov \left[ \langle f_1, \theta \rangle_{\tilde{H}_R}, \langle f_2, \theta \rangle_{\tilde{H}_R} \right] &= \langle f_1, f_2 \rangle_{H_R} \quad \forall f_1, f_2 \in H_R\end{aligned}\tag{1.60}$$

Therefore, any random variable  $z(t)$  which is a linear function of  $\{\theta(s), s \in S\}$  can be expressed as  $z(t) = \langle g(t, \cdot), \theta \rangle_{\tilde{H}_R}$  for some  $g(t, \cdot) \in H_R$ , and it is the representer of  $z(t)$ , that is, it is obtained from  $\{R(\cdot, s), s \in S\}$  by the same operations as  $z(t)$  is obtained from  $\{\theta(s), s \in S\}$ .

As we can see the RKHS defined by the Gaussian kernel and the RKHS defined by the covariance function are very different from each other. The eigenfunctions  $\Phi_i(x)$  of the kernel nonlinearly map an input sample into a functional in a high-dimensional space, as needed for better representation (regression, classification). The RKHS  $H_R$  (defined by the covariance kernel) on the other hand, maps linearly entire random vectors into a scalar function (i.e., a point). Statistical inference in  $H_R$  can be done with simple algebra as Parzen and others have elucidated. The mapping functions  $\Phi_i(x)$  of  $H_\kappa$  are independent of the data statistics, whereas the mapping functions  $\Phi_i(x)$  for  $H_R$  incorporates naturally the data statistics due to the expected value operator. Moreover, the elements of the Gram matrix of  $H_R$  are deterministic, whereas the elements of the Gram matrix of  $H_\kappa$  are stochastic when the input is a random variable. Therefore, one sees how different RKHS can be, depending upon how the elements of the space and the kernel are defined, which illustrates how powerful the methodology really is.

### Optimal Filtering Problem in RKHS $H_R$

Given a time series  $\{Y(t), t \in T\}$  with an underlying signal and additive noise  $Y(t) = X(t) + N(t)$ ,  $0 \leq t \leq T$  and a random variable  $Z$  (which can be the value of  $X$  in the past or in the future, or any other random variable) find for each  $t \in T$  the best random variable in  $L_2(Y(t), t \in T)$  whose mean square distance to  $Z$  is minimum. In probability theory this can be accomplished by finding the conditional expectation of the random variable  $Z$  relative to a  $\sigma$  field defined by  $X(t)$ . Wiener developed an important alternate solution using spectral factorization [333], and Kolmogorov provided an equivalent and elegant geometric solution [185]. An alternative result derived by Parzen [238] from orthogonal projections and direct sums of elements of complementary Hilbert spaces provides the existence and uniqueness of the same basic solution in a RKHS. We will briefly present this solution here.

Let the wide-sense conditional mean of  $X(t)$  given the entire set of  $Y(\cdot)$  be

$$X(t) = E^p[X(t) | Y(s), s \in T]$$

where the superscript  $p$  denotes projection in the Hilbert space. The wide sense conditional mean can be posed as the following optimization problem. Let  $H_Y = H(Y(t), t \in T)$  be the Hilbert space of the square integrable random variables spanned by the family  $\{Y(t), t \in T\}$ . Find  $\hat{U}$  in  $H_Y$  such that

$$E \left[ |\hat{U} - X(t)|^2 \right] = \min_{U \in H_Y} E \left[ |U - X(t)|^2 \right] \quad (1.61)$$

The vector  $\hat{U}$  will be denoted as  $\hat{X}(t)$ . Parzen showed it can be obtained as the solution of the normal equations

$$E[\hat{X}(t)Y(s)] = E[X(t)Y(s)], \quad s \in T \quad (1.62)$$

which can be written as

$$\hat{X}(t) = \langle E[X(t)Y(s)], Y(s) \rangle_{\tilde{H}_{E[Y(t1)Y(t2)]}}$$

In fact, define the covariance kernel on  $T \times T$   $K_Y(t_1, t_2) = E[Y(t_1), Y(t_2)]$ . Let  $H_{K_Y}$  be the RKHS corresponding to  $K_Y$ , then for each  $t \in T$ ,  $K_{XY}(t, s) = E[Y(t), Y(s)] \in H_{K_Y}$  and  $\hat{X}(\cdot) = \langle K_{XY}(\cdot, s), Y(s) \rangle_{\tilde{H}_{K_Y}}$ . For uncorrelated signal and noise, let us represent  $R(t_1, t_2) = E[N(t_1), N(t_2)]$  and  $K(t_1, t_2) = E[X(t_1), X(t_2)]$ , then  $K_Y = R + K$ ,  $K_{XY} = K$  and

$$\hat{X}(\cdot) = \langle K(\cdot, s), Y(s) \rangle_{\tilde{H}_{R+K}} \quad (1.63)$$

Eq. (1.63) is the equivalent of the Wiener–Hopf solution for the optimal filtering problem, but obtained in the RKHS.

In order to help draw the similarity, let us write loosely the optimal projection as  $\hat{X}(t) = \int_T W(t, u)Y(u)du$ , and with this notation, the normal equations of Eq. (1.62) can be rewritten as

$$\hat{X}(t) = \int_T W(t, u)[K(u, s) + R(u, s)]du = K(t, s), \quad s \in T \quad (1.64)$$

which need to be solved for the projector  $W(t, u)$ . Define the operator  $R + K$  on  $L_2(T)$  as follows:  $(R + K)f$  is the function whose value at  $s$  is

$$\{(R + K)f\}(s) = \int_T f(u)\{K(u, s) + R(u, s)\}du$$

With this notation the solution to Eq. (1.64) becomes  $W(t, u) = \{(R + K)^{-1}K(t, \cdot)\}(u)$ , and substituting in Eq. (1.64) yields  $\hat{X}(t) = \int_T Y(u)\{(R + K)^{-1}K(t, \cdot)\}du$  for which Eq. (1.63) is a rigorous form.

## 1.12 Conclusions

This chapter gave an overview of the foundations for the approach developed in the remainder of the book. The apparently unrelated topics of information theory, optimal filtering, and RKHS theory can be easily linked from the point of view of the expected value of the PDF operator, which we call the information potential, the argument of the logarithm of Renyi's quadratic entropy. When new synergisms are established the hope is that they can lead to new advances and later be exploited for statistical signal processing, machine learning, and information-theoretic applications.

The chapter starts with a brief review of the problem of optimal transmission of information in communication systems and how information theory concepts were developed to answer them. The review emphasizes understanding of the fundamental concepts of entropy and mutual information and how they have been put to good use in the design of communication systems. Extensions to the original definitions of entropy and mutual information proposed by Shannon are also discussed to provide a glimpse of how the original theory evolved. The role of information theoretic concepts in machine learning is also briefly reviewed to point out the role of mesoscopic descriptors of the data that can be used for new cost functions and new learning principles based on information-theoretic concepts.

The foundations of adaptive filter theory (which are remarkably similar to regression in discrete time) are also presented. The impact of information-theoretic cost functions (entropy and divergences) in regression, filtering, classification and unsupervised learning is presented. Basically, information-theoretic costs unify supervised and unsupervised algorithms in the sense that changing the inputs to the cost function yields both classes of learning paradigms.

The last part of Chapter 1 deals with basic definitions of reproducing kernel Hilbert spaces. The review also illustrates how powerful the RKHS methodology is because alternate definitions of the domain and range of the functional mapping provides radically different RKHS characteristics. Basically, we can say that there are RKHS for representation and for statistical inference. The review also demonstrates the importance of RKHS for adaptive filtering, by showing how the LMS algorithm can be easily ported to RKHS defined by the Gaussian kernel function. The uses of the autocorrelation function of a random process as an alternative kernel is also discussed to show the flexibility of the RKHS framework and how it can be used for statistical inference.

# Renyi's Entropy, Divergence and Their Nonparametric Estimators

Dongxin Xu and Deniz Erdogmuns

## 2.1 Introduction

It is evident from Chapter 1 that Shannon's entropy occupies a central role in information-theoretic studies. Yet, the concept of information is so rich that perhaps there is no single definition that will be able to quantify information properly. Moreover, from an engineering perspective, one must estimate entropy from data which is a nontrivial matter. In this book we concentrate on Alfred Renyi's seminal work on information theory to derive a set of estimators to apply entropy and divergence as cost functions in adaptation and learning. Therefore, we are mainly interested in computationally simple, nonparametric estimators that are continuous and differentiable in terms of the samples to yield well-behaved gradient algorithms that can optimize adaptive system parameters. There are many factors that affect the determination of the optimum of the performance surface, such as gradient noise, learning rates, and misadjustment, therefore in these types of applications the entropy estimator's bias and variance are not as critical as, for instance, in coding or rate distortion theories. Moreover in adaptation one is only interested in the extremum (maximum or minimum) of the cost, with creates independence from its actual values, because only relative assessments are necessary. Following our nonparametric goals, what matters most in learning is to develop cost functions or divergence measures that can be derived directly from data without further assumptions to capture as much structure as possible within the data's probability density function (PDF).

The chapter starts with a review of Renyi's entropy origins, its properties, and interpretations. Then a new estimator for Renyi's quadratic entropy is developed using kernel density estimation. With cost functions for adaptation in mind, the properties of this estimator which is called the *Information Potential (IP) estimator* are carefully presented, including its bias and variance. A physical interpretation of the IP is presented which will motivate new adaptation algorithms in Chapter 3.

A brief review of Renyi's divergence and mutual information is also presented, and two divergence measures in probability spaces are introduced that have the great appeal of being computed from combinations of the IP estimator; that is, IP can be readily extended to estimate divergences. A detailed discussion of the algorithms and interpretations of these divergence measures is presented to allow their use in practical applications. This includes two classes of algorithms that speed up the computations to  $O(N)$ . Furthermore, Appendix A presents a review of entropy estimators along with a review of how the IP can be used in practical problems.

## 2.2 Definition and Interpretation of Renyi's Entropy

The parametric family of entropies was introduced by Alfred Renyi in the mid 1950s as a mathematical generalization of Shannon entropy [263]. Renyi wanted to find the most general class of information measure that preserved the additivity of statistically independent systems and were compatible with Kolmogorov's probability axioms.

Let us assume a discrete probability distribution  $P = \{p_1, p_2, \dots, p_N\}$  fulfilling the conditions of  $\sum_k p_k = 1$ ,  $p_k \geq 0$ . If one observes the outcome of two independent events with probabilities  $p$  and  $q$ , additivity of information for independent events requires that the corresponding information  $I(\cdot)$  obey Cauchy's functional equation (i.e. the information of the joint event is the sum of the information of each event)

$$I(P \cdot Q) = I(P) + I(Q). \quad (2.1)$$

Therefore, the amount of information produced by knowing that an event with probability  $p$  took place could be, apart from a multiplicative factor (normalized by setting  $I(1/2) = 1$ )

$$I(P) = -\log_2 p, \quad (2.2)$$

which is similar to Hartley's amount of information. Let us further assume that the outcomes of some experimental discrete random variable occur with probabilities  $p_1, \dots, p_N$ , and if the  $k$ th outcome delivers  $I_k$  bits of information then the total amount of information for the set  $\Gamma = \{I_1, \dots, I_N\}$  is

$$I(P) = \sum_{k=1}^N p_k I_k \quad (2.3)$$

which can be recognized as Shannon's entropy  $H(X)$ . However, we have assumed the linear averaging operator in this formulation. In the general theory of means for any monotonic function  $g(x)$  with an inverse  $g^{-1}(x)$  one

can define the general mean associated with  $g(x)$  for a set of real values  $\{x_k, k = 1, \dots, N\}$  with probabilities of  $\{p_k\}$  as

$$g^{-1} \left( \sum_{k=1}^N p_k g(x_k) \right).$$

Applying this definition to the information  $I(P)$ , we obtain

$$I(P) = g^{-1} \left( \sum_{k=1}^N p_k g(I_k) \right), \quad (2.4)$$

where  $g(x)$  is a Kolmogorov–Nagumo invertible function [229]. This  $g(x)$  is the so called quasi-linear mean and it constitutes the most general mean compatible with Kolmogorov's axiomatics [184]. Renyi then proved that when the postulate of additivity for independent events is applied to Eq. (2.4) it dramatically restricts the class of possible  $g(x)$ . In fact, only two classes are possible;  $g(x) = cx$  with  $c$  a constant, which states that for linear  $g(x)$  the quasi-linear mean reduces to the ordinary mean and yields the Shannon information measure Eq.(2.3). Hence, Shannon's information is the averaged information in the usual sense, and becomes the simplest of the information measures. The other functional class is  $g(x) = c 2^{(1-\alpha)x}$  which implies

$$I_\alpha(P) = \frac{1}{1-\alpha} \log \left( \sum_{k=1}^N p_k^\alpha \right)$$

with  $\alpha \neq 1$  and  $\alpha \geq 0$ , and it is called Renyi's information measure of order  $\alpha$ , or Renyi's  $\alpha$  entropy, denoted as  $H_\alpha(X)$ . We adopt the term “entropy” since Renyi showed that it also represents the disclosed information (or removed ignorance) after analyzing the expression in a close analogy with Shannon's theory.

At a first glance, the main difference between Shannon and Renyi's entropies is the placement of the logarithm in the expression. In Shannon entropy (Eq. (1.4), the probability mass function (PMF) weights the  $\log(p_k)$  term, whereas in Renyi's entropy the log is outside a term that involves the  $\alpha$  power of the PMF. In order to compare further with Shannon's entropy let us rewrite Renyi's entropy as

$$\begin{aligned} H_\alpha(X) &= \frac{1}{1-\alpha} \log \left( \sum_{k=1}^N p_k^\alpha \right) = -\log \left( \sum_{k=1}^N p_k^\alpha \right)^{\frac{1}{\alpha-1}} \\ &= -\log \left( \sum_{k=1}^N p_k p_k^{\alpha-1} \right)^{\frac{1}{\alpha-1}}. \end{aligned} \quad (2.5)$$

We see in Eq. (2.5) that the PMF  $p_k$  also weights a term that now is the  $(\alpha - 1)$  power of the probability mass function. Let us denote the argument

of the log as  $V_\alpha(X) = \sum_k kp_k^\alpha = E[p_k^{\alpha-1}]$  which is called in this book the  $\alpha$  *information potential* ( $IP_\alpha$ ) and allows rewriting Eq. (2.5) as

$$H_\alpha(X) = \frac{1}{1-\alpha} \log(V_\alpha(X)) = -\log(\sqrt[\alpha-1]{V_\alpha(X)}). \quad (2.6)$$

At a deeper level, Renyi's entropy measure is much more flexible due to the parameter  $\alpha$ , enabling several measurements of uncertainty (or dissimilarity) within a given distribution [177]. Considered as a function of  $\alpha$ ,  $H_\alpha(X)$  is normally called the spectrum of Renyi information and its graphical plot is useful in statistical inference [308]. The value at  $\alpha = 1$  is particularly important because it provides the expected value of the negative log-likelihood ( $E[-\log p(x)]$ ) while its derivative with respect to  $\alpha$  is proportional to the variance of the log-likelihood function ( $\dot{H}_1(X) = -1/2\text{var}(\log p(x))$ ). Due to this fact it is possible to derive an index of the intrinsic shape of the PDF as  $S(X) = -2\dot{H}_1(X)$  which has more statistical power than kurtosis and can be used as a partial order for the tails of distributions.

To find the most fundamental (and possibly irreducible) set of properties characterizing Renyi's information it is desirable to axiomatize it. Various axiomatizations have been proposed [1, 265]. For our purpose the most convenient set of axioms is the following [340].

1. The entropy measure  $H(p_1, \dots, p_N)$  is a continuous function of all the probabilities  $p_k$ , which means that a small change in probability distribution will only result in a small change in the entropy.
2.  $H(p_1, \dots, p_N)$  is permutationally symmetric; that is, the position change of any two or more  $p_k$  in  $H(p_1, \dots, p_N)$  will not change the entropy value. Actually, the permutation of any  $p_k$  in the distribution will not change the uncertainty or disorder of the distribution and thus should not affect the entropy.
3.  $H(1/n, \dots, 1/n)$  is a monotonic increasing function of  $n$ . For an equiprobable distribution, when the number of choices increases, the uncertainty or disorder increases, and so does the entropy measure.
4. Recursivity: If an entropy measure satisfies

$$\begin{aligned} H(p_1, p_2, \dots, p_N) &= H(p_1 + p_2, p_3, \dots, p_N) \\ &\quad + (p_1 + p_2)^\alpha H\left(\frac{p_1}{p_1 + p_2}, \frac{p_2}{p_1 + p_2}\right) \end{aligned}$$

then it has a recursivity property. It means that the entropy of  $N$  outcomes can be expressed in terms of the entropy of  $N - 1$  outcomes plus the weighted entropy of the combined two outcomes.

5. Additivity: If  $p = (p_1, \dots, p_N)$  and  $q = (q_1, \dots, q_N)$  are two independent probability distributions, and the joint probability distribution is denoted by  $p \cdot q$ , then the property  $H(p \cdot q) = H(p) + H(q)$  is called additivity.

The table compares Renyi's entropy property versus Shannon for these axioms.

Properties	(1)	(2)	(3)	(4)	(5)
Shannon	yes	yes	yes	yes	yes
Renyi	yes	yes	yes	no	yes

Notice that Renyi's recursivity property differs from Shannon's recursivity, so we entered *no* in Property (4) to make this fact clear. Further properties of Renyi's entropy were studied extensively in [1, 265]. We list here a few of the key ones.

- (a)  $H_\alpha(X)$  is nonnegative:  $H_\alpha(X) \geq 0$ .
- (b)  $H_\alpha(X)$  is decisive:  $H_\alpha(0, 1) = H_\alpha(1, 0) = 0$ .

For  $\alpha \leq 1$  Renyi's entropy is concave. For  $\alpha > 1$  Renyi's entropy is neither pure convex nor pure concave. It loses concavity for  $\alpha > \alpha^* > 1$  where  $\alpha^*$  depends on  $N$  as  $\alpha^* \leq 1 + \ln(4)/\ln(N - 1)$ .

- (d) Because

$$\frac{\alpha - 1}{\alpha} H_\alpha(X) \leq \frac{\beta - 1}{\beta} H_\beta(X), \quad \alpha \geq \beta,$$

- $(\alpha - 1)H_\alpha(X)$  is a concave function of  $p_k$ .
- (e)  $H_\alpha(X)$  is a bounded, continuous and nonincreasing function of  $\alpha$ .
- (f) For  $\alpha \in R$ ;  $H_\alpha(A \cap B) = H_\alpha(A) - H_\alpha(B|A)$  with  $H_\alpha(B|A) = g^{-1}(\sum_k \rho_k(\alpha)g(H_\alpha(B|A = A_k)))$ , which can be interpreted as the conditional entropy with  $\rho_k(\alpha) = p_k^\alpha / \sum_k p_k^\alpha$  and  $g$  an invertible and positive function in  $[0,1)$ .
- (g)  $H_z(X)$  with  $z = \alpha + j\omega$  is analytic in the entire complex plane except the negative real axis. Therefore the singularity at  $\alpha = 1$  is not essential and we obtain  $\lim_{\alpha \rightarrow 1} H_\alpha(X) = H_S(X)$ .

The ambiguous concavity property of Renyi's entropy in (b) makes it incompatible with the requirements of physical entropy (unlike Shannon's) when expressed as a function of the pertinent  $p_k$ . The implications of (g) are far reaching. It can be shown that if we perform an analytical continuation of  $\alpha$  in the complex domain (e.g.,  $z = \alpha + j\omega$ ),  $H_z(X) = \sum_{k=1}^N p_k^z$  is analytic except in the negative real axis. More specifically, if we make  $z = 1 + re^{j\omega}$ ,  $H_z(X)$  is analytic in the interior of the circle of radius  $r$  so it is also analytic at  $z = \alpha = 1$ . Therefore Renyi's entropy is differentiable at  $z = 1$  to all orders. With this proof, Shannon entropy can be uniquely determined from the behavior of (analytically continued) Renyi's entropy in the vicinity of  $z = 1$ . Therefore from a strict mathematical point of view, Shannon entropy is not a special information measure deserving separate axiomatization but a member of Renyi's wide class of entropies embraced by a single unifying axiomatic [168]. Despite its formal origin Renyi's entropy proved important in a variety of practical applications: coding theory [44], statistical inference [236], quantum mechanics (as an

estimator for von Neumann entropy) [32], chaotic dynamical systems [120], multifractal analysis [168], and as a measure of diversity in economics [135].

### Geometric Interpretation of Renyi's Entropy

Before we actually start the derivation of the estimators, we investigate further the role of  $\alpha$  by providing a geometric picture of Renyi's entropy that is very useful to describe this family of entropies. Probability mass functions can be visualized geometrically as points in a vector space called the simplex with the axis given by the random variables. The simplex  $\Delta_N$  consists of all possible probability distributions for an  $N$  multidimensional random variable; that is,

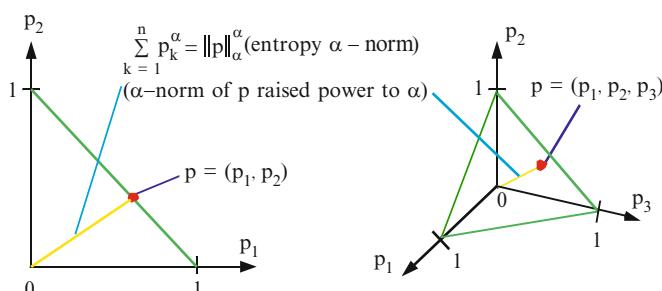
$$\Delta_N = \left\{ p = (p_1, \dots, p_N)^T \in R^N, p_i \geq 0, \sum_i p_i = 1, \forall i \right\}$$

For instance, for three variables ( $x, y, z$ ), the space of all such distributions is an equilateral triangle with vertices at  $(1,0,0)$ ,  $(0,1,0)$ ,  $(0,0,1)$  (a convex subset of  $R^3$ ). Figure 2.1 shows the simplex for  $N = 2$  and  $N = 3$ .

Any point in the simplex is a different PMF and has a different distance to the origin. If one defines the PMF  $\alpha$ -norm as

$$\|p(x)\|_\alpha = \sqrt[\alpha]{\sum_{k=1}^N p_k^\alpha} = \sqrt[\alpha]{V_\alpha(X)},$$

that is, the  $\alpha$ -information potential  $V_\alpha(x)$  can be interpreted as the  $\alpha$  power of the PMF  $\alpha$ -norm. Specifically, Renyi's  $\alpha$  entropy takes the  $\alpha - 1$  root of  $V_\alpha(x)$  and rescales it by the negative of the logarithm as specified in Eq. (2.6). Therefore  $\alpha$  specifies in the simplex the norm to measure the distance of  $p(x)$  to the origin. As is well known from the theory of norms [40], the free parameter  $\alpha$  specifying the norm changes the importance of small values versus large values in the set. Three  $\alpha$  cases are of special interest:  $H_0$  is the logarithm of the number of nonzero components of the distribution and is known



**Fig. 2.1.** The simplex for  $N = 2, 3$  and the entropy  $\alpha$  norm.

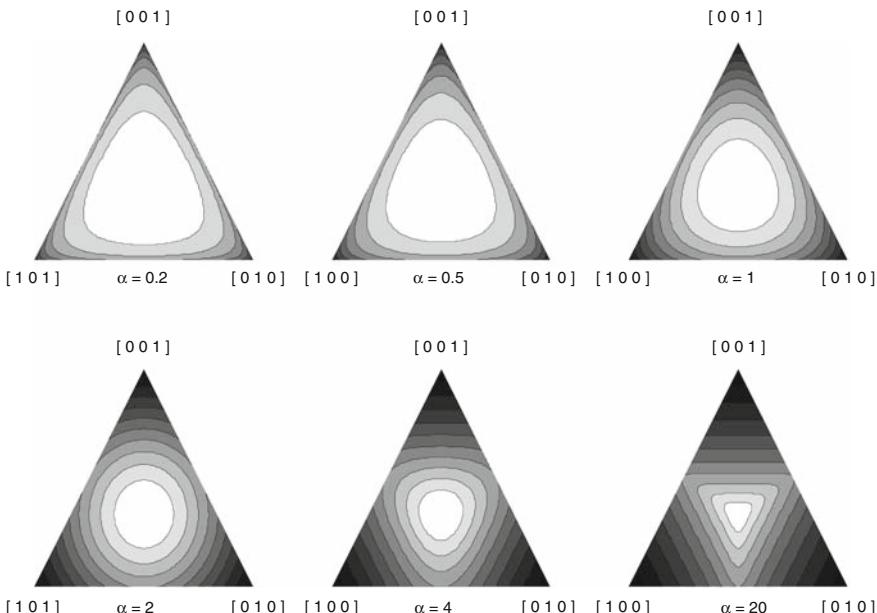
as Hartley's entropy.  $H_\infty$  can be thought of as  $\lim_{\alpha \rightarrow \infty} H_{R^\alpha} = H_\infty$  with  $H_\infty = -\log(\max_k(p_k))$  which is called the Chebyshev entropy [177]. The most interesting special case is obtained for  $\lim_{\alpha \rightarrow 1} H_\alpha = H_S$  which means that Shannon's entropy is the limiting case of the 1-norm of the probability mass function  $p(x)$ . Actually, the 1-norm of any probability density is always 1 by definition, which will give 0/0 in Eq (2.6). Using the l'Hôpital rule we can proceed and evaluate Eq. (2.6) as

$$\lim_{\alpha \rightarrow 1} H_\alpha(X) = \lim_{\alpha \rightarrow 1} \frac{\frac{d}{d\alpha} \log \sum_{k=1}^N p_k^\alpha}{\frac{d}{d\alpha}(1-\alpha)} = \frac{\left( \sum_{k=1}^N \log p_k \cdot p_k^\alpha \right) \left( \sum_{k=1}^N p_k^\alpha \right)^{-1}}{-1} \Big|_{\alpha=1} = H_S(X) \quad (2.7)$$

so, in the limit, Shannon's entropy can be regarded as the functional value of the 1-norm of the probability density.

Renyi's entropy is a scalar that characterizes densities, thus it is also interesting to display the contours of equal Renyi's entropy in the simplex (Figure 2.2) for several  $\alpha$ .

In order to illustrate how Renyi's entropy evaluation behaves in the simplex, we plot the isoentropy contours as a function of  $\alpha$ . Notice that for  $\alpha$  close to zero the values inside the simplex change very little, and the Shannon case basically preserves the shape of these contours except that there is a more visible change. Observe that for  $\alpha = 2$  the contours are circular, meaning a



**Fig. 2.2.** Isoentropy contour in the  $N = 3$  probability simplex for different  $\alpha$  values.

$\ell_2$ -norm to the center. For higher values of  $\alpha$  the contours rotate by 180 degrees and emphasize changes with respect to the central point when  $\alpha$  increases.

When  $\alpha > 1$ , Renyi's entropy  $H_\alpha$  are monotonic decreasing functions of  $\text{IP}_\alpha$ . So, in this case, the entropy maximization is equivalent to IP minimization, and the entropy minimization is equivalent to IP maximization.

When  $\alpha \leq 1$ , Renyi's entropy  $H_\alpha$  are monotonic increasing functions of the  $V_\alpha$ . So, in this case, the entropy maximization is equivalent to IP maximization, and the entropy minimization is equivalent to IP minimization.

### Renyi's Quadratic Entropy $H_2$

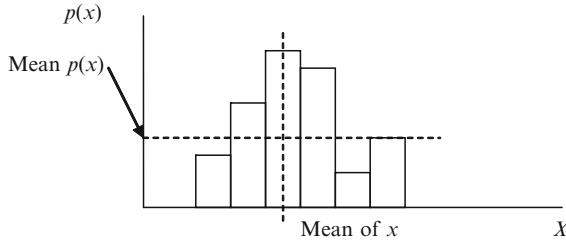
$H_2$  is of particular interest in this book and it is a monotonic decreasing function of the  $\alpha = 2$  information potential  $V_2$  ( $V$  for short) of the PMF  $p(x)$ .  $H_2$  implicitly uses an Euclidean distance from the point  $p(x)$  in the simplex to the origin of the space.

$$H_2(X) = -\log \left( \sum_k p_k^2 \right). \quad (2.8)$$

In the particle physics literature, the second moment of the probability mass function is known as the index of coincidence or *purity* (because it vanishes if the state of the particle is pure) [32]. The *linear entropy* is defined as  $H_L(X) = 1 - p^2(x)$  (which is in fact the Havrda-Charvat [138] or Tsallis entropy of second order [320]), but in Renyi's case, the logarithm is used instead. In econometrics, Renyi's quadratic entropy has been used to quantify diversity [135]. Because  $H_2$  is a lower bound of Shannon's entropy, it might be more efficient than Shannon's entropy for entropy maximization.

One aspect that we would like to stress after the presentation of the geometric picture of Renyi's entropy is the fact that the argument of the log in Renyi's quadratic entropy,  $V_2 = E[p(x)]$  has meaning in itself as the expected value of the PMF. Equivalently, if one considers the PMF a nonlinear function of the random variable  $x$  and defines the transformed random variable  $\xi = p(x)$ ,  $\text{IP}_2$  (IP for short) becomes the expected value of  $\xi$ . The argument of the log in  $H_2(x)$  is central to our studies. In fact, we show that in optimization (parameter adaptation) the logarithm is irrelevant (because it is a monotonic function and therefore does not affect the location of the extremum of the cost function in the space of the system parameters) and is dropped almost from the beginning of our adaptive system studies. This is unthinkable in communication theory, because there the fundamental issue is the additivity of information, which is intrinsically linked to the logarithmic function.

Some authors [32] define  $f_\alpha(\mathbf{p}) = \sum_{k=1}^N p_k^\alpha$  as the  $\alpha$  *moment of the probability mass function*, which is Schur concave for  $\alpha < 1$  and Schur convex for  $\alpha > 1$ . Therefore Renyi's entropy is a function of the moment of the vector variable  $\mathbf{p} = [p_1, p_2, \dots, p_N]$ . Moreover, the moments  $f_\alpha(p_1, p_2, \dots, p_N)$  for  $\alpha = 2, \dots, N$  define the vector  $\mathbf{p}$  up to a permutation of its components, which means that the spectrum of Renyi's entropies defines the probability



**Fig. 2.3.** Relation between mean of  $x$  and mean of  $p(x)$ .

mass function in a similar manner as the characteristic function expansion. The  $\alpha$  moments also relate Renyi's to von Neumann's entropy [326]. It is important not to confuse the moments of the PMF with the moments of the data, therefore we prefer to use the information potential terminology, which also has a powerful analogy as we discuss later. Figure 2.3 shows the relation between the 2-norm of the PMF (mean of  $\xi$ ) and the mean of the data, which should be obvious.

### Renyi's Entropy of Continuous Variables

Renyi's entropy can also be defined for continuous random variables. Let  $p(x)$  be the continuous PDF defined in  $[0,1]$ . The integrated probability is

$$p_{n,k} = \int_{k/n}^{(k+1)/n} p(x)dx, \quad k = 0, 1, \dots, n-1$$

and by defining the discrete mass function  $P_n = \{p_{n,k}\}$  it is possible to show [265] that

$$H_\alpha(X) = \lim_{n \rightarrow \infty} (I_\alpha(P_n) - \log n) = \frac{1}{1-\alpha} \log \int p^\alpha(x)dx. \quad (2.9)$$

This is very similar to the Shannon case, showing that the differential Renyi's entropy can be negative for  $\alpha \leq 1$ . Indeed  $\log(n)$  can be thought as the entropy of the uniform distribution, and so the continuous entropy is the gain obtained by substituting the uniform distribution by the experimental samples  $P_n$ . The generalization for multidimensions proceeds along the same arguments, preserving the functional form of Eq. (2.9). Quadratic Renyi's entropy for continuous random variables reads

$$H_2(X) = -\log \int p^2(x)dx. \quad (2.10)$$

We use capital  $H$  for differential entropy throughout this book.

## 2.3 Quadratic Renyi's Entropy Estimator

As already stated in Chapter 1, in experimental science, one is faced with the issue of estimating entropy directly from samples in a nonparametric way because it is often not prudent to advance with a parametric PDF model. In such cases we have to resort to a nonparametric estimation. But instead of first estimating the PDF and then computing its entropy, here we seek the direct approach of estimating quadratic Renyi's entropy from samples by estimating  $E[p(X)]$ , which is a scalar. In adaptive systems we are mostly interested in continuous random variables, and this is the case on which we concentrate from this point on.

Recall the definition of quadratic entropy given in Eq. (2.10) for the continuous random variable  $X$ . Suppose we have  $N$  independent and identically distributed (i.i.d.) samples  $\{x_1, \dots, x_N\}$  from this random variable. The kernel (Parzen) estimate of the PDF [241] using an arbitrary kernel function  $\kappa_\sigma(\cdot)$  is given by

$$\hat{p}_X(x) = \frac{1}{N\sigma} \sum_{i=1}^N \kappa\left(\frac{x - x_i}{\sigma}\right) \quad (2.11)$$

where  $\sigma$  is the kernel size or bandwidth parameter. This kernel function has to obey the following properties [300].

1.  $\kappa(x) \geq 0$ .
2.  $\int_R \kappa(x) dx = 1$ .
3.  $\lim_{x \rightarrow \infty} |x\kappa(x)| = 0$ .

Normally one uses a symmetric normalized kernel that peaks at the sample and for our purposes it must be continuous and differentiable (reasons are discussed later). Kernel density estimation is a well-studied topic and the use of kernels has been widespread since the seminal work of Rosenblatt and Parzen. The quality of estimators is normally quantified by their bias and variance, and for kernel estimation they are respectively given by [241] ( $\hat{\cdot}$  denotes estimated quantities)

$$\begin{aligned} Bias(\hat{p}_\sigma(x)) &= E[\hat{p}_\sigma(x)] - p(x) \approx \sigma^2 / 2p''(x)\mu(K) \\ Var(\hat{p}_\sigma(x)) &= E[(\hat{p}_\sigma(x) - E[\hat{p}_\sigma(x)])^2] \approx \frac{1}{N\sigma} \|K\|_2^2 p(x), \quad N\sigma \rightarrow \infty \end{aligned} \quad (2.12)$$

where  $\mu(K)$  and  $\|K\|^2$  are constants given by the specific kernel utilized, and  $p''$  is the second derivative of the PDF. As one can see in Eq. (2.12) the kernel size affects the bias and the variance in opposite ways, so the best kernel size is a compromise between bias and variance of the estimator. It is well known from Parzen's seminal work [241] that the class of kernel estimators is asymptotically unbiased when the kernel size tends to zero (i.e., the kernel approaches a Dirac delta function), and consistent in quadratic mean when

the number of samples increases to infinite (the product  $N\sigma$  must tend to infinite). Moreover, one can show that the mean square error between the true and estimated PDF can decrease for the optimal kernel size at a rate as high as  $N^{-4/5}$  for scalar variables, which is close to the best possible ( $1/N$ ). For symmetric kernels such as the Gaussian it is typically  $N^{-2/5}$ .

The difficulty of density estimation, in particular in high dimensions and with few samples, is that one wants to obtain a reasonable estimate for all the points in the domain. This is an ill-posed problem (see Appendix A). However, for  $V_2(X)$ , we are only interested in estimating a single number  $E[p(x)]$ . Assuming Gaussian kernels Eq. (1.50),  $G_\sigma(\cdot)$ , with standard deviation  $\sigma$  and substituting this in the quadratic entropy expression Eq. (2.10), we get after straightforward substitutions the estimator

$$\begin{aligned}\hat{H}_2(X) &= -\log \int_{-\infty}^{\infty} \left( \frac{1}{N} \sum_{i=1}^N G_\sigma(x - x_i) \right)^2 dx \\ &= -\log \frac{1}{N^2} \int_{-\infty}^{\infty} \left( \sum_{i=1}^N \sum_{j=1}^N G_\sigma(x - x_j) \cdot G_\sigma(x - x_i) \right) dx \\ &= -\log \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \int_{-\infty}^{\infty} G_\sigma(x - x_j) \cdot G_\sigma(x - x_i) dx \\ &= -\log \left( \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N G_{\sigma\sqrt{2}}(x_j - x_i) \right).\end{aligned}\quad (2.13)$$

The result is easily obtained by noticing that the integral of the product of two Gaussians is *exactly evaluated* as the value of the Gaussian computed at the difference of the arguments and whose variance is the sum of the variances of the two original Gaussian functions. Other kernel functions, however, do not result in such convenient evaluation of the integral because the Gaussian maintains the functional form under convolution. Nevertheless, any positive definite function that peaks at the origin (most kernels) might still be used in the estimation, but the expressions become a bit more complicated. We named the argument of the log in Eq. (2.13) (i.e., the kernel estimator of the 2-norm of the PMF (or PDF)) the *quadratic information potential estimator* (simply IP when there is no confusion) for reasons that become apparent later.

## Information Potential for Entropy Estimation

The argument of the logarithm in quadratic Renyi's entropy that has been called the information potential can be estimated directly from data as

$$\hat{H}_2(X) = -\log(\hat{V}_2(X)) \quad \hat{V}_{2,\sigma}(X) = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N G_{\sigma\sqrt{2}}(x_j - x_i) \quad (2.14)$$

where  $\hat{V}_{2,\sigma}(X)$  is the quadratic IP estimator that depends on  $\sigma$ . Let us compare this with the conventional way that entropy is estimated from data. In practical cases, the estimation of Shannon or Renyi's entropy directly from data will follow the route:

$$\text{data} \rightarrow \text{pdf estimation} \rightarrow \text{integral estimation}.$$

Notice that entropy is a scalar, but as an intermediate step one has to estimate a function (the PDF), which is much harder in high dimensional spaces. With quadratic Renyi's entropy and the IP (i.e.  $V(X)$ ) we bypassed the explicit need to estimate the PDF; that is, the calculations follow the path

$$\text{data} \rightarrow \text{IP}(\hat{V}_2(X)) \rightarrow \text{algebra}.$$

Eq. (2.14) is one of the central results of information-theoretic learning because it shows that the Information Potential, which is a scalar, can be estimated directly from samples with an exact evaluation of the integral over the random variable for Gaussian kernels. Eq. (2.14) shows that the IP is only a function of sample pairs, instead of the PDF shape. This is similar to the conventional estimators of the mean and the variance that work directly with the samples irrespective of the PDF, but unfortunately here the estimator has a free parameter and it shares the properties of kernel density estimation.

There are two important implications of Eq. (2.14). As is apparent, the variance of the Gaussian (also called the kernel size or bandwidth) is a free parameter that needs to be selected by the user. Therefore, when the IP is estimated, the resulting values of entropy depend on the kernel size selected, which is also a crucial problem in density estimation [300]. The estimated values of the IP have little absolute meaning due to this kernel size dependence, but it gauges performance in a relative sense when comparing data generated with the same set of parameters. In learning (the main purpose of this book) the system parameters depend only on the cost function's extremum location in parameter space, not of the cost's actual value, so the IP dependence on kernel size is more manageable than for applications that require the actual value of the estimated quantity.

The way we interpret the kernel bandwidth is as a *scale parameter* for the analysis. It has to be selected according to the data dynamic range and number of samples to make the estimation of the entropy meaningful. Silverman's rule [300] is

$$\sigma_{opt} = \sigma_x \left( 4N^{-1}(2d+1)^{-1} \right)^{\frac{1}{(d+4)}}, \quad (2.15)$$

where  $N$  is the number of samples,  $d$  is the data dimensionality, and  $\sigma_x$  is the data standard deviation. Although Eq. (2.15) was derived for Gaussian distributions it is sufficient for most of our applications. The bandwidth selection is treated more thoroughly in Appendix A. To summarize, we want to say that the existence of this free parameter is a double-sided sword: it provides

flexibility in the application of the methodology to real data; but on the other hand it either requires a selection criterion or a scanning over  $\sigma$  because the effect of the kernel size is much harder to quantify than the scale in wavelet decompositions or frequency in Fourier analysis that also contain a free parameter in their definitions. More generally, it shows the functional nature of entropy estimation using kernels.

The second implication is that the estimator is  $O(N^2)$ , which may create computation bottlenecks for large datasets. This is the price we have to pay to estimate entropy with the IP when compared with mean and variance. Indeed, both the mean and the variance estimators work with a single sample at a time (in fact the variance also requires pairwise computation but one of the elements of the pair is the mean that can be computed a priori), but if we are interested in qualifying the “shape” of the PDF with Renyi’s second-order entropy, pairwise interactions are necessary. We show later in this chapter how the fast Gauss transform and the incomplete Cholesky decomposition solve this problem with algorithms that are  $O(N)$ .

### Extended Estimator for $\alpha$ -Renyi’s Entropy

It turns out that the pairwise interaction model can be generalized from an estimator of Renyi’s quadratic entropy to all  $\alpha \neq 1$ . In essence  $\hat{H}_2$  is the centerpiece for nonparametric kernel estimation of Renyi’s entropy as we show below. Consider the definition of Renyi’s order- $\alpha$  entropy in Eq. (2.9), which can also be written with an expectation operator as

$$H_\alpha(X) \triangleq \frac{1}{1-\alpha} \log \int_{-\infty}^{\infty} p_X^\alpha(x) dx = \frac{1}{1-\alpha} \log E_X [p_X^{\alpha-1}(X)]. \quad (2.16)$$

Approximating the expectation operator with the sample mean as is commonly done in density estimation [300], we get

$$H_\alpha(X) \approx \hat{H}_\alpha(X) = \frac{1}{1-\alpha} \log \frac{1}{N} \sum_{j=1}^N p_X^{\alpha-1}(x_j). \quad (2.17)$$

Notice that we never had to address this approximation in deriving Eq. (2.14), therefore we can expect that an estimator of Eq. (2.17) will differ from  $\hat{H}_2(X)$  in Eq. (2.13), i.e. it will have different bias and variance. Finally, substituting the Parzen window estimator of Eq. (2.11) in Eq. (2.17) and rearranging terms, we obtain a nonparametric plug-in estimator for Renyi’s  $\alpha$  entropy as

$$\hat{H}_\alpha(X) = \frac{1}{1-\alpha} \log \frac{1}{N} \sum_{j=1}^N \left( \frac{1}{N} \sum_{i=1}^N \kappa_\sigma(x_j - x_i) \right)^{\alpha-1} = \frac{1}{1-\alpha} \log(\hat{V}_{\alpha,\sigma}(X)), \quad (2.18)$$

where the  $\alpha$  information potential estimator (the dependence on  $\sigma$  is normally omitted)

$$\hat{V}_{\alpha,\sigma}(X) = \frac{1}{N^\alpha} \sum_{j=1}^N \left( \sum_{i=1}^N \kappa_\sigma(x_j - x_i) \right)^{\alpha-1}.$$

The nonparametric estimator in Eq. (2.18) can still be written as the log of the  $\alpha$ -norm of  $\hat{V}_\alpha(X) = IP_\alpha(X)$ , but again it differs from the IP of Eq. (2.14). For  $\alpha = 1$ ,  $\hat{V}_1 = 1$  for any PDF. For all  $\alpha \geq 0, \alpha \neq 1$  it is a general-purpose estimator and can be used to evaluate  $\alpha$  entropy directly from samples or to adapt the weights of a learning system based on an entropic performance index. We now study its properties in detail.

## 2.4 Properties of Renyi's Nonparametric Entropy Estimators

In the following, all kernel functions and random variable samples are assumed to be single-dimensional unless noted otherwise. The generalization of these results to multidimensional cases is trivial and the proofs follow similar lines. We start by analyzing the accuracy of the approximation of the expected value by the sample average.

**Property 2.1.** For the special case of Gaussian kernels, the estimator  $\hat{V}_\alpha(X)$  of Eq. (2.18) only differs from the  $\hat{V}_2(X)$  of Eq. (2.14) by a factor of  $\sqrt{2}$  in the kernel size.

*Proof.* A direct comparison proves the property. This difference stems from the need to approximate the expected value by the sample mean in Eq. (2.18). In fact, the estimator of Eq. (2.18) requires two approximations, the approximation of the expected value by the sample mean and the kernel approximation of the PDF, whereas Eq. (2.13) only requires the kernel approximation of the PDF. Therefore in general they yield two different estimators of the same statistical quantity. However, what is interesting is that for  $\alpha = 2$  the sample mean approximation for finite  $N$  and Gaussian kernels can still be exactly compensated by a change of the kernel size from  $\sigma$  to  $\sigma\sqrt{2}$  in Eq. (2.18).

**Property 2.2.** For any kernel function  $\kappa(x)$  that obeys the relation

$$\kappa^{new}(x_j - x_i) = \int_{-\infty}^{\infty} \kappa^{old}(x - x_i) \cdot \kappa^{old}(x - x_j) dx, \quad (2.19)$$

where  $\kappa^{new}(\cdot)$  denotes the kernel function used in Eq. (2.18) and  $\kappa^{old}(\cdot)$  denotes the kernel function used in Eq. (2.14), the estimator of Renyi's quadratic entropy of Eq. (2.18) matches the estimator of Eq. (2.14) using the IP.

*Proof.* Direct substitution proves the property. These properties reiterate the privileged place of the Gaussian kernel and quadratic Renyi's entropy in ITL. The case  $\alpha = 2$  also allows a very interesting link between ITL and kernel learning, and explains the reason why the sample mean approximation is not necessary in IP computation.

It is possible to show using the properties of the Gaussian kernel that Renyi's  $\alpha$  entropy estimator can be written exactly as

$$\hat{H}_\alpha(X) = \frac{1}{1-\alpha} \log \left\{ \frac{1}{N^\alpha} \left( \frac{1}{\sqrt{2\pi}\sigma} \right)^{\alpha-1} \left( \frac{1}{\sqrt{\alpha}} \right) \left( \sqrt{2\pi}\sqrt{\alpha\sigma} \right)^{\binom{\alpha}{2}} \sum_{i_1=1}^N \dots \sum_{i_\alpha=1}^N \left[ \prod_{p=1}^{\alpha} \prod_{q=1, q>p}^{\alpha} G_{\sigma\sqrt{\alpha}}(x_{i_p} - x_{i_q}) \right] \right\} \quad (2.20)$$

or

$$\hat{H}_\alpha(X) = \frac{1}{1-\alpha} \log \left\{ \frac{1}{N^\alpha} \left( \frac{1}{\sqrt{2\pi}\sigma} \right)^{\alpha-2} \sum_{i_1=1}^N \dots \sum_{i_\alpha=1}^N G_{\sqrt{\alpha}\sigma} \left( \sum_{p=1}^{\alpha} \sum_{q=1, q>p}^{\alpha} (x_{i_p} - x_{i_q})^2 \right) \right\}. \quad (2.21)$$

In either form one still sees the kernel size for the Gaussian being multiplied by  $\sigma\sqrt{\alpha}$  as could be expected from Property 2.1, however, these expressions are not easily compared with Eq. (2.18) even when the kernel is a Gaussian. Therefore, the practical estimation of  $\alpha$  Renyi's entropy with kernels will follow the approximation of the expected value by the sample mean as indicated in Eq. (2.18).

**Property 2.3.** The kernel size must be a parameter that satisfies the scaling property  $\kappa_{c\sigma}(x) = \kappa_\sigma(x/c)/c$  for any positive factor  $c$  [241].

This regulatory condition guarantees that changes in the kernel size can be compensated by linear scaling in the domain and range of the estimated quantities. In the analysis of the eigenstructure of the entropy cost function near the global optimum and in obtaining scale-invariant entropy-based cost functions, this property becomes useful.

**Property 2.4.** The entropy estimator in Eq. (2.18) is invariant to the mean of the underlying density of the samples as is the actual entropy [86].

*Proof.* Consider two random variables  $X$  and  $\bar{X}$  where  $\bar{X} = X + m$  with  $m$  being a real number. The entropy of  $\bar{X}$  becomes

$$\begin{aligned} H_\alpha(\bar{X}) &= \frac{1}{1-\alpha} \log \int p_{\bar{X}}^\alpha(\bar{x}) d\bar{x} = \frac{1}{1-\alpha} \log \int p_X^\alpha(\bar{x} - m) d\bar{x} \\ &= \frac{1}{1-\alpha} \log \int p_X^\alpha(x) dx = H_\alpha(X). \end{aligned} \quad (2.22)$$

Let  $\{x_1, \dots, x_N\}$  be samples of  $X$ , then samples of  $\bar{X}$  are  $\{x_1 + m, \dots, x_N + m\}$ . Therefore, the estimated entropy of  $\bar{X}$  is

$$\begin{aligned} \hat{H}_\alpha(\bar{X}) &= \frac{1}{1-\alpha} \log \frac{1}{N^\alpha} \sum_{j=1}^N \left( \sum_{i=1}^N \kappa_\sigma(\bar{x}_j - \bar{x}_i) \right)^{\alpha-1} \\ &= \frac{1}{1-\alpha} \log \frac{1}{N^\alpha} \sum_{j=1}^N \left( \sum_{i=1}^N \kappa_\sigma(x_j + m - x_i - m) \right)^{\alpha-1} = \hat{H}_\alpha(X). \end{aligned} \quad (2.23)$$

Due to this property of the entropy and its estimator, when the entropy cost is utilized in supervised learning the mean of the error signal is not necessarily zero, which is a requirement for most applications. This requirement has to be implemented by adding a bias term to the system output that makes the mean of the error equal to zero. Because of this feature, entropy does not define a metric in the space of the samples. We address this point in more detail in Chapter 3. However, when we are interested in the statistical properties of the signals other than their means, this is not a problem.

**Property 2.5.** The limit of Renyi's entropy as  $\alpha \rightarrow 1$  is Shannon's entropy. The limit of the entropy estimator in Eq. (2.18) as  $\alpha \rightarrow 1$  is Shannon's entropy estimated using Parzen windowing with the expectation approximated by the sample mean.

*Proof.* Notice that Renyi's entropy in Eq. (2.16) is discontinuous at  $\alpha = 1$ . However, when we take its limit as this parameter approaches one, we get Shannon's entropy as shown in Eq. (2.24) for continuous variables (similarly to Eq. (2.8)),

$$\begin{aligned} \lim_{\alpha \rightarrow 1} H_\alpha(X) &= \lim_{\alpha \rightarrow 1} \frac{1}{1-\alpha} \log \int p_X^\alpha(x) dx \\ &= \frac{\lim_{\alpha \rightarrow 1} \int \log p_X(x) \cdot p_X^\alpha(x) dx / \int p_X^\alpha(x) dx}{\lim_{\alpha \rightarrow 1} -1} \\ &= - \int p_X(x) \cdot \log p_X(x) dx = H_S(X) \end{aligned} \quad (2.24)$$

The derivation of this result for the estimator in Eq. (2.18) is shown in Eq. (2.25).

$$\begin{aligned}
\lim_{\alpha \rightarrow 1} \hat{H}_\alpha(X) &= \lim_{\alpha \rightarrow 1} \frac{1}{1-\alpha} \log \frac{1}{N} \sum_{j=1}^N \left( \frac{1}{N} \sum_{i=1}^N \kappa_\sigma(x_j - x_i) \right)^{\alpha-1} \\
&= - \lim_{\alpha \rightarrow 1} \frac{\left( \frac{1}{N} \sum_{j=1}^N \left( \frac{1}{N} \sum_{i=1}^N \kappa_\sigma(x_j - x_i) \right)^{\alpha-1} \log \left( \frac{1}{N} \sum_{i=1}^N \kappa_\sigma(x_j - x_i) \right) \right)}{\left( \frac{1}{N} \sum_{j=1}^N \left( \frac{1}{N} \sum_{i=1}^N \kappa_\sigma(x_j - x_i) \right)^{\alpha-1} \right)} \\
&= \frac{-1}{N} \sum_{j=1}^N \log \left( \frac{1}{N} \sum_{i=1}^N \kappa_\sigma(x_j - x_i) \right) = \hat{H}_S(X).
\end{aligned} \tag{2.25}$$

In terms of adaptation, this means that all the conclusions drawn in this research about Renyi's entropy, its estimator, and training algorithms based on Renyi's entropy apply in the limit of  $\alpha \rightarrow 1$ , to Shannon's definition as well.

**Property 2.6.** In order to maintain consistency with the scaling property of the actual entropy, if the entropy estimate of samples  $\{x_1, \dots, x_N\}$  of a random variable  $X$  is estimated using a kernel size of  $\sigma$ , the entropy estimate of the samples  $\{cx_1, \dots, cx_N\}$  of a random variable  $cX$  must be estimated using a kernel size of  $|c|\sigma$ .

*Proof.* Consider the Renyi's entropy of the random variable  $cX$ , whose PDF is  $p_X(x/c)/|c|$  in terms of the PDF of the random variable  $X$  and the scaling coefficient  $c$ .

$$H_\alpha(cX) = \frac{1}{1-\alpha} \log \int_{-\infty}^{\infty} \frac{1}{|c|^\alpha} p_X^\alpha \left( \frac{x}{c} \right) dx = H_\alpha(X) + \log |c|. \tag{2.26}$$

Now consider the entropy estimate of the samples  $\{cx_1, \dots, cx_N\}$  using the kernel size  $|c|\sigma$ .

$$\begin{aligned}
\hat{H}_\alpha(cx) &= \frac{1}{1-\alpha} \log \frac{1}{N^\alpha} \sum_{j=1}^N \left( \sum_{i=1}^N \kappa_{|c|\sigma} \left( \frac{x_j - x_i}{c} \right) \right)^{\alpha-1} \\
&= \frac{1}{1-\alpha} \log \frac{1}{N^\alpha} \sum_{j=1}^N \left( \sum_{i=1}^N \frac{1}{|c|} \kappa_\sigma \left( \frac{cx_j - cx_i}{c} \right) \right)^{\alpha-1} \\
&= \frac{1}{1-\alpha} \log \frac{1}{|c|^{\alpha-1} N^\alpha} \sum_{j=1}^N \left( \sum_{i=1}^N \kappa_\sigma(x_j - x_i) \right)^{\alpha-1} \\
&= \hat{H}_\alpha(X) + \log |c|.
\end{aligned} \tag{2.27}$$

This property is crucial when the problem requires a scale-invariant cost function as in blind deconvolution illustrated in Chapter 8. The scaling of the kernel size as described above according to the norm of the weight vector guarantees that the nonparametric estimation of the scale-invariant cost function possesses this property as well.

**Property 2.7.** When estimating the joint entropy of an  $n$ -dimensional random vector  $X$  from its samples  $\{x_1, \dots, x_N\}$ , use a multidimensional kernel that is the product of single-dimensional kernels. In this way, the estimate of the joint entropy and estimate of the marginal entropies are consistent.

*Proof.* Let the random variable  $X_o$  be the  $o$ th component of  $X$ . Consider the use of single-dimensional kernels  $\kappa_{\sigma_o}(\cdot)$  for each of these components. Also assume that the multidimensional kernel used to estimate the joint PDF of  $X$  is  $\kappa_{\Sigma}(\cdot)$ . The Parzen estimate of the joint PDF is then given by

$$\hat{p}_X(x) = \frac{1}{N} \sum_{i=1}^N \kappa_{\Sigma}(x - x_i). \quad (2.28)$$

Similarly, the Parzen estimate of the marginal density of  $X_o$  is

$$\hat{p}_{X,o}(x) = \frac{1}{N} \sum_{i=1}^N \kappa_{\sigma_o}(x_o - x_o(i)). \quad (2.29)$$

Without loss of generality, consider the marginal PDF of  $X_1$  derived from the estimate of the joint PDF in Eq. (2.28).

$$\begin{aligned} \hat{p}_{X,1}(x_1) &= \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \bar{p}_X(x_1, \dots, x_n) dx_2, \dots, dx_n \\ &= \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \frac{1}{N} \sum_{i=1}^N \kappa_{\Sigma}(x_1 - x_1(i), \dots, x_n - x_n(i)) dx_2, \dots, dx_n \\ &= \frac{1}{N} \sum_{i=1}^N \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \kappa_{\Sigma}(x_1 - x_1(i), \dots, x_n - x_n(i)) dx_2, \dots, dx_n. \end{aligned} \quad (2.30)$$

Now, assuming that the joint kernel is the product of the marginal kernels evaluated at the appropriate values (i.e.,  $\kappa_{\Sigma}(x) = \prod_{o=1}^N \kappa_{\sigma_o}(x_o)$ ), we get Eq. (2.31). Thus, this choice of the multidimensional kernel for joint entropy estimation guarantees consistency between the joint and marginal PDF and entropy estimates. This property is, in fact, critical for the general PDF estimation problem besides being important in entropy estimation.

$$\begin{aligned}
p_{X,1}(x_1) &= \frac{1}{N} \sum_{i=1}^N \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \prod_{o=1}^n \kappa_{\sigma_o}(x_o - x_o(i)) dx_2, \dots, dx_n \\
&= \frac{1}{N} \sum_{i=1}^N \kappa_{\sigma_1}(x_1 - x_1(i)) \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \prod_{o=2}^n \kappa_{\sigma_o}(x_o - x_o(i)) dx_2, \dots, dx_n \\
&= \frac{1}{N} \sum_{i=1}^N \kappa_{\sigma_1}(x_1 - x_1(i)) \left( \prod_{o=2}^n \int_{-\infty}^{\infty} \kappa_{\sigma_o}(x_o - x_o(i)) dx^o \right) = \hat{p}_{X,1}(x_1).
\end{aligned} \tag{2.31}$$

This important issue should be considered in adaptation scenarios where the marginal entropies of multiple signals and their joint entropy are used in the cost function simultaneously. It is desirable to have consistency between the marginal and joint entropy estimates.

**Theorem 2.1.** *The entropy estimator in Eq. (2.18) is consistent if the Parzen windowing and the sample mean are consistent for the actual PDF of the i.i.d. samples.*

*Proof.* The proof follows immediately from the consistency of the Parzen window estimate for the PDF and the fact that as  $N$  goes to infinity the sample mean converges to the expected value which makes Eq. (2.18) approach Eq. (2.16) (e.g., the sample mean estimate is not consistent for infinite-variance PDFs).

This theorem is important because it points out the asymptotic limitations of the estimator. In adaptation and learning from finite samples, because we rarely have huge datasets, consistency is not the primary issue, but the bias and variance of the estimator must still be known. Their effect in the location of the extremum of the function in the space of the parameters is the real issue.

**Theorem 2.2.** *If the maximum value of the kernel  $\kappa_\sigma(\xi)$  is achieved when  $\xi = 0$ , then the minimum value of the entropy estimator in Eq. (2.18) is achieved when all samples are equal to each other, that is,  $x_1 = \dots = x_N = c$  [86].*

*Proof.* By substitution, we find that the entropy estimator takes the value  $-\log \kappa_\sigma(0)$  when all samples are equal to each other. We need to show that

$$\frac{1}{1-\alpha} \log \frac{1}{N^\alpha} \sum_{j=1}^N \left( \sum_{i=1}^N \kappa_\sigma(x_j - x_i) \right)^{\alpha-1} \geq -\log \kappa_\sigma(0). \tag{2.32}$$

For  $\alpha > 1$ , this is equivalent to showing that

$$\sum_{j=1}^N \left( \sum_{i=1}^N \kappa_\sigma(x_j - x_i) \right)^{\alpha-1} \leq N^\alpha \kappa_\sigma^{\alpha-1}(0). \quad (2.33)$$

Replacing the left-hand side of Eq. (2.33) with its upper bound we get Eq. (2.34). Because the kernel function is chosen such that its maximum occurs when its argument is zero, we obtain the desired result given in Eq. (2.33).

$$\begin{aligned} \sum_{j=1}^N \left( \sum_{i=1}^N \kappa_\sigma(x_j - x_i) \right)^{\alpha-1} &\leq N \max_j \left[ \left( \sum_{i=1}^N \kappa_\sigma(x_j - x_i) \right)^{\alpha-1} \right] \\ &\leq N \max_j \left[ N^{\alpha-1} \max_i \kappa_\sigma^{\alpha-1}(x_j - x_i) \right] = N^\alpha \max_{i,j} \kappa_\sigma^{\alpha-1}(x_j - x_i). \end{aligned} \quad (2.34)$$

The proof for the case  $\alpha < 1$  is similar. It uses the min operator instead of max due to the direction of the inequality.

In supervised training, it is imperative that the cost function achieve its global minimum when all the error samples are zero. Minimum error entropy learning using this entropy estimator, which is introduced in Chapter 3, becomes a valid supervised training approach with this property of the entropy estimator. In addition, the unsupervised training scenarios such as minimum entropy blind deconvolution, which are discussed in Chapter 8, benefit from this property of the estimator as well.

**Theorem 2.3.** *If the kernel function  $\kappa_\sigma(\cdot)$  is continuous, differentiable, symmetric, and unimodal, then the global minimum described in Theorem 2.2 of the entropy estimator in Eq. (2.18) is smooth, that is, it has a zero gradient and a positive semidefinite Hessian matrix.*

*Proof.* Let  $\bar{\mathbf{x}} = [x_1, \dots, x_N]^T$  be the data samples collected in a vector for notational simplicity. Without loss of generality, consider the dataset given by  $\bar{\mathbf{x}} = 0$ , meaning all samples are zero. With some algebra, the gradient and the Hessian matrix of the expression in Eq. (2.18) with respect to  $\bar{\mathbf{x}}$  are found as

$$\begin{aligned} \frac{\partial \hat{H}_\alpha}{\partial x_k} &= \frac{1}{1-\alpha} \frac{\partial \hat{V}_\alpha / \partial x_k}{\hat{V}_\alpha} \\ \frac{\partial^2 \hat{H}_\alpha}{\partial x_l \partial x_k} &= \frac{1}{1-\alpha} \frac{(\partial^2 \hat{V}_\alpha / \partial x_l \partial x_k) \hat{V}_\alpha - (\partial \hat{V}_\alpha / \partial x_k)(\partial \hat{V}_\alpha / \partial x_l)}{\hat{V}_\alpha^2}. \end{aligned} \quad (2.35)$$

where the variable  $\hat{V}_\alpha$  is the argument of the logarithm in the final expression in Eq. (2.18). Evaluating these expressions at  $\bar{\mathbf{x}} = 0$ , which corresponds to the maximum value of the kernel we get

$$\begin{aligned}
\hat{V}_\alpha \Big|_{\bar{x}=0} &= \kappa_\sigma^{\alpha-1}(0) \\
\frac{\partial \hat{V}_\alpha}{\partial x_k} \Big|_{\bar{x}=0} &= \frac{(\alpha-1)}{N^\alpha} [N^{\alpha-1} \kappa_\sigma^{\alpha-2}(0) \kappa'(0) - N^{\alpha-1} \kappa_\sigma^{\alpha-2}(0) \kappa'(0)] = 0 \\
\frac{\partial^2 \hat{V}_\alpha}{\partial x_k^2} \Big|_{\bar{x}=0} &= \frac{(\alpha-1)(N-1) \kappa_\sigma^{\alpha-3}(0)}{N^2} [(\alpha-2) \kappa'^2(0) + 2\kappa(0) \kappa''(0)] \\
\frac{\partial^2 \hat{V}_\alpha}{\partial x_l \partial x_k} \Big|_{\bar{x}=0} &= -\frac{(\alpha-1) \kappa_\sigma^{\alpha-3}(0)}{N^2} [(\alpha-2) \kappa'^2(0) + 2\kappa(0) \kappa''(0)], \quad (2.36)
\end{aligned}$$

which shows that the gradient vector is zero and that the Hessian matrix is composed of

$$\frac{\partial^2 \hat{H}_\alpha}{\partial x_l \partial x_k} \Big|_{\bar{x}=0} = \begin{cases} -(N-1) \kappa_\sigma^{-\alpha-1}(0) [(\alpha-2) \kappa'^2(0) + 2\kappa(0) \kappa''(0)] / N^2, & l = k \\ \kappa_\sigma^{-\alpha-1}(0) [(\alpha-2) \kappa'^2(0) + 2\kappa(0) \kappa''(0)] / N^2, & l \neq k \end{cases} \quad (2.37)$$

Denoting the diagonal terms by  $a$  and the off-diagonal terms by  $b$ , we can determine all the eigenvalue-eigenvector pairs of this matrix to be

$$\begin{aligned}
&\{0, [1, \dots, 1]^T\}, \{aN/(N-1), [1, -1, 0, \dots, 0]^T\}, \{aN/(N-1), \\
&\quad [1, 0, -1, 0, \dots, 0]^T\}, \dots
\end{aligned}$$

Notice that the nonzero eigenvalue has a multiplicity of  $N-1$  and for a kernel function as described in the theorem and for  $N > 1$  this eigenvalue is positive, because the kernel evaluated at zero is positive, the first derivative of the kernel evaluated at zero is zero, and the second derivative is negative. Thus the Hessian matrix at the global minimum of the entropy estimator is negative semidefinite. This is to be expected because there is one eigenvector corresponding to the direction that only changes the mean of data, along which the entropy estimator is constant due to Property 2.4.

In adaptation using numerical optimization techniques, it is crucial that the global optimum be a smooth point in the weight space with zero gradient and finite-eigenvalue Hessian. This last theorem shows that the nonparametric estimator is suitable for entropy minimization adaptation scenarios.

**Property 2.8.** If the kernel function satisfies the conditions in Theorem 2.3, then in the limit, as the kernel size tends to infinity, the quadratic entropy estimator approaches the logarithm of a scaled and biased version of the sample variance.

*Proof.* Let  $\{x_1, \dots, x_N\}$  be the samples of  $X$ . We denote the second-order sample moment and the sample mean with the following.

$$\bar{x}^2 = \frac{1}{N} \sum_{j=1}^N x_j^2 \quad \bar{x}^2 = \left( \frac{1}{N} \sum_{j=1}^N x_j \right)^2.$$

By assumption the kernel size is very large, therefore the pairwise differences of samples will be very small compared to the kernel size, thus allowing the second-order Taylor series expansion of the kernel function around zero to be a valid approximation. Also, due to the kernel function being symmetric and differentiable, its first-order derivative at zero will be zero yielding

$$\kappa_\sigma(\xi) \approx \kappa_\sigma(0) + \kappa'_\sigma(0)\xi + \kappa''_\sigma(0)\xi^2/2 = \kappa_\sigma(0) + \kappa''_\sigma(0)\xi^2/2. \quad (2.38)$$

Substituting this in the quadratic entropy estimator obtained from Eq. (2.18) by substituting  $\alpha = 2$ , we get Eq. (2.39), where  $\bar{x}^2 - \bar{x}^2$  is the sample variance. Notice that the kernel size affects the scale factor multiplying the sample variance in Eq. (2.39). In addition to this, there is a bias depending on the kernel's center value.

$$\begin{aligned} \hat{H}_2(X) &\approx -\log \left[ \frac{1}{N^2} \sum_{j=1}^N \sum_{i=1}^N (\kappa_\sigma(0) + \kappa''_\sigma(0)(x_j - x_i)^2/2) \right] \\ &= -\log \left[ \kappa_\sigma(0) + \frac{1}{2}\kappa''_\sigma(0) \left( \frac{1}{N^2} \sum_{j=1}^N \sum_{i=1}^N (x_j^2 - 2x_j x_i + x_i^2) \right) \right] \\ &= -\log \left[ \kappa_\sigma(0) + \frac{1}{2}\kappa''_\sigma(0) (\bar{x}^2 - \bar{x}^2) \right]. \end{aligned} \quad (2.39)$$

**Property 2.9.** In the case of joint entropy estimation, if the multidimensional kernel function satisfies  $\kappa_\Sigma(\xi) = \kappa_\Sigma(R^{-1}\xi)$  for all orthonormal matrices  $R$ , then the entropy estimator in Eq. 2.18 is invariant under rotations as is the actual entropy of a random vector  $X$ . Notice that the condition on the joint kernel function requires hyperspherical symmetry.

*Proof.* Consider two  $n$ -dimensional random vectors  $X$  and  $\bar{X}$  related to each other with  $\bar{X} = RX$  where  $R$  is an  $n \times n$  real orthonormal matrix. Then the entropy of  $\bar{X}$  is

$$\begin{aligned} H_\alpha(\bar{X}) &= \frac{1}{1-\alpha} \log \int_{-\infty}^{\infty} p_{\bar{X}}^\alpha(\bar{x}) d\bar{x} = \frac{1}{1-\alpha} \log \int_{-\infty}^{\infty} \frac{1}{|R|^\alpha} p_X^\alpha(R^{-1}\bar{x}) d\bar{x} \\ &= \frac{1}{1-\alpha} \log \int_{-\infty}^{\infty} \frac{1}{|R|^\alpha} p_X^\alpha(x) |R| dx = \frac{1}{1-\alpha} \log |R|^{1-\alpha} \int_{-\infty}^{\infty} p_X^\alpha(x) dx \\ &= H_\alpha(X) + \log |R| = H_\alpha(X). \end{aligned} \quad (2.40)$$

Now consider the estimation of the joint entropy of  $\bar{X}$  from its samples, which are given by  $\{Rx_1, \dots, Rx_N\}$ , where  $\{x_1, \dots, x_N\}$  are samples of  $X$ . Suppose we use a multidimensional kernel  $\kappa_\Sigma(\cdot)$  that satisfies the required condition. This results in Eq. (2.41). In adaptation scenarios where the invariance-under-rotations property of entropy needs to be exploited, the careful choice of the joint kernel becomes important. Property 2.5 describes how to select kernel functions in such situations.

$$\begin{aligned}
\hat{H}_\alpha(\bar{X}) &= \frac{1}{1-\alpha} \log \frac{1}{N^\alpha} \sum_{j=1}^N \left( \sum_{i=1}^N \kappa_\Sigma(Rx_j - Rx_i) \right)^{\alpha-1} \\
&= \frac{1}{1-\alpha} \log \frac{1}{N^\alpha} \sum_{j=1}^N \left( \sum_{i=1}^N \frac{1}{|R|} \kappa_\Sigma(R^{-1}(Rx_j - Rx_i)) \right)^{\alpha-1} \\
&= \frac{1}{1-\alpha} \log |R|^{\alpha-1} \frac{1}{N^\alpha} \sum_{j=1}^N \left( \sum_{i=1}^N \kappa_\Sigma(x_j - x_i) \right)^{\alpha-1} \\
&= \hat{H}_\alpha(X)
\end{aligned} \tag{2.41}$$

**Theorem 2.4.**  $\lim_{N \rightarrow \infty} \hat{H}_\alpha(X) = H_\alpha(\hat{X}) \geq H_\alpha(X)$ , where  $\hat{X}$  is a random variable with the PDF  $f_X(\cdot)^* \kappa_\sigma(\cdot)$ . The equality occurs if and only if the kernel size is zero. This result is also valid on the average for the finite-sample case.

*Proof.* It is well known that the Parzen window estimate of the PDF of  $X$  converges consistently to  $f_X(\cdot)^* \kappa_\sigma(\cdot)$ . Therefore, the entropy estimator in Eq. (2.18) converges to the actual entropy of this PDF. To prove the inequality consider

$$e^{(1-\alpha)H_\alpha(\hat{X})} = \int_{-\infty}^{\infty} p_{\hat{X}}^\alpha(y) dy = \int_{-\infty}^{\infty} \left[ \int_{-\infty}^{\infty} \kappa_\sigma(\tau) p_X^\alpha(y - \tau) d\tau \right] dy. \tag{2.42}$$

Using Jensen's inequality for convex and concave cases, we get Eq. (2.43), where we defined the mean-invariant quantity  $V_\alpha(X)$  as the integral of the  $\alpha$ th power of the PDF of  $X$ , which is the argument of the log in the definition of Renyi's entropy given in Eq. (2.16). Reorganizing the terms in Eq. (2.43) and using the relationship between entropy and information potential, regardless of the value of  $\alpha$  and the direction of the inequality, we arrive at the conclusion  $H_\alpha(\hat{X}) \geq H_\alpha(X)$ . The fact that these results are also valid on the average for the finite-sample case is due to the property  $E[\hat{p}_X(\cdot)] = p_X(\cdot)^* \kappa_\sigma(\cdot)$  of Parzen windowing, which relates the average PDF estimate to the actual value and the kernel function.

$$\begin{aligned}
\exp((1-\alpha)H_\alpha(\hat{X})) &\stackrel{\alpha \geq 1}{\leq} \left( \int_{-\infty}^{\infty} \left[ \int_{-\infty}^{\infty} \kappa_\sigma(\tau) [p_X(y - \tau)]^\alpha d\tau \right] dy \right) \\
&= \int_{-\infty}^{\infty} \kappa_\sigma(\tau) \left[ \int_{-\infty}^{\infty} [p_X(y - \tau)]^\alpha dy \right] d\tau \\
&= \int_{-\infty}^{\infty} \kappa_\sigma(\tau) V_\alpha(X) d\tau \\
&= V_\alpha(X) \cdot \int_{-\infty}^{\infty} \kappa_\sigma(\tau) d\tau = V_\alpha(X).
\end{aligned} \tag{2.43}$$

This theorem is useful in proving asymptotic noise rejection properties of the entropy-based adaptation criteria, and shows that for entropy minimization, the proposed estimator provides a useful approximation in the form of an upper bound to the true entropy of the signal under consideration.

## 2.5 Bias and Variance of the Information Potential Estimator

### IP Estimator Bias

In this section, the bias of the information potential is analyzed for finite samples using the shape of the data probability density function (which is unknown for most cases, but provides understanding of the factors involved). We call the attention of the readers to the analysis of the density estimation in Appendix A, which should be used to contrast the results obtained in this section. The same basic approach is taken here, but a simplified notation will be used and some steps are omitted. We choose the Gaussian kernel for density estimation. The information potential estimator is

$$\hat{V}_{2,\sigma}(X) = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N G_\sigma(x_j - x_i). \quad (2.44)$$

The IP bias is obtained by taking the expectation

$$\begin{aligned} E[\hat{V}_2(X)] &= E \left[ \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N G_\sigma(x_j - x_i) \right] \\ &= \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N E[G_\sigma(x_j - x_i)] = E[G_\sigma(x_j - x_i)]. \end{aligned} \quad (2.45)$$

Now expanding the PDF in Taylor series and using the i.i.d. assumption on the data and the definition of expected value

$$\begin{aligned} E[G_\sigma(x_j - x_i)] &= \iint G_\sigma(x_i - x_j) p(x_i) p(x_j) dx_i dx_j \\ &= \int p(x_i) \left[ \int G_\sigma(s) p(x_i + \sigma s) ds \right] dx_i \quad s = (x_i - x_j)/\sigma \\ &= \int p(x) \left\{ \int G_\sigma(s) [p(x) + \sigma s p'(x) + 1/2(\sigma^2 s^2 p''(x) \right. \\ &\quad \left. + o(\sigma^4)] ds \right\} dx \\ &\approx \int p(x) [p(x) + 1/2(\sigma^2 p''(x)) \mu_2(G)] dx \quad \text{as } \sigma \rightarrow 0 \\ &= \int p^2(x) dx + (\sigma^2/2) \mu_2(G) \int p(x) p''(x) dx. \end{aligned} \quad (2.46)$$

where for the Gaussian kernel  $\mu_2(G) = \int \sigma^2 G_\sigma(s) ds = 1$ . Now

$$\int p(x)p''(x)dx = E[p''(X)] \quad (2.47)$$

and combining the above equations we obtain

$$Bias[\hat{V}_2(X)] = E[\hat{V}_2(X)] - \int p^2(x)dx = (\sigma^2/2)E[p''(X)] \quad (2.48)$$

We see that the bias of IP for Gaussian kernels increases proportionally to the square of the kernel size multiplied by the expected value of the PDF second derivative. This result has the same basic form of Eq. (2.12) for kernel density estimation (see Appendix A), except that the double derivative of the PDF is substituted by its expected value, which is better behaved.

### IP Estimator Variance

To analyze the variance of the information potential, we rewrite Eq. (2.44) as

$$N^2\hat{V}(X) = \sum_i \sum_j G_\sigma(x_i - x_j)$$

and take the variance of both sides to obtain:

$$\begin{aligned} N^4 Var(\hat{V}(X)) &= N^4 \left[ E(\hat{V}^2(X)) - (E(\hat{V}(X)))^2 \right] \\ &= \sum_i \sum_j \sum_k \sum_l \{ E[G_\sigma(x_i - x_j)G_\sigma(x_k - x_l)] \\ &\quad - E[G_\sigma(x_i - x_j)] E[G_\sigma(x_k - x_l)] \} \end{aligned} \quad (2.49)$$

The right-hand side of Eq. (2.49) consists of  $N^4$  terms. These terms can be classified into four categories according to the possible values of  $i, j, k$ , and  $l$ :

1. If  $i, j, k, l$  are all different among each other, then, according to the independence assumption, their joint distribution can be factorized as

$$E[G_\sigma(x_i - x_j)G_\sigma(x_k - x_l)] = E[G_\sigma(x_i - x_j)] E[G_\sigma(x_k - x_l)] \quad (2.50)$$

therefore, all the positive terms in the summation cancel out the corresponding negative terms.

2. If  $j \neq i = k \neq l$  and  $j \neq l$ , then,  $x_j, x_l$  would be independent when  $x_i = x_k$  and those terms can be calculated as

$$E[G_\sigma(x_i - x_j)G_\sigma(x_l - x_i)] - E[G_\sigma(x_i - x_j)] E[G_\sigma(x_k - x_l)], \quad (2.51)$$

Choosing different values of  $i, j$ , and  $k$  there are totally  $N(N - 1)(N - 2)$  terms like Eq. (2.51).

3. If  $i = k \neq j = l$ , by the same argument, there are  $N(N - 1)$  terms and these terms are all equal:

$$\begin{aligned} & E [G_\sigma(x_i - x_j)G_\sigma(x_j - x_i)] - E [G_\sigma(x_i - x_j)] E [G_\sigma(x_j - x_i)] \\ &= E [G_\sigma(x_i - x_j)^2] - E [G_\sigma(x_i - x_j)]^2 = \text{Var}[G_\sigma(x_i - x_j)]. \end{aligned} \quad (2.52)$$

4. If  $i = k = j \neq l$ , then, similarly, the independent terms can be written as:

$$\begin{aligned} & E [G_\sigma(x_i - x_i)G_\sigma(x_i - x_l)] - E [G_\sigma(x_i - x_i)] E [G_\sigma(x_i - x_l)] \\ &= E [G_\sigma(0)G_\sigma(x_i - x_l)] - E [G_\sigma(0)G_\sigma(x_i - x_l)] = 0. \end{aligned} \quad (2.53)$$

From this discussion, we see that only Cases 2 and 3 will yield a nonzero value, and they will affect the variance of the information potential with different weights; that is, since the number of terms in Case 2 is  $N(N - 1)(N - 2)$  which is proportional to  $N^3$  while the number of terms in Case 3 is  $N(N - 1)$  which is proportional to  $N^2$ . Thus, as the number of sample  $N$  increases, Eq. (2.51) becomes dominant. We denote

$$\begin{aligned} a &= E [K_\sigma(x_i - x_j)K_\sigma(x_j - x_l)] - E [K_\sigma(x_i - x_j)] E [K_\sigma(x_j - x_l)] \\ b &= \text{Var}[K_\sigma(x_i - x_j)], \end{aligned} \quad (2.54)$$

where  $K$  is the Gaussian kernel of Eq. (1.50) without the division by  $\sigma$ . If  $a \neq 0$  which is generally true for most probability density functions we can write:

$$\begin{aligned} \text{Var}(\hat{V}(X)) &= E(\hat{V}^2(X)) - (E(\hat{V}(X)))^2 \\ &= \frac{aN(N - 1)(N - 2) + bN(N - 1)}{\sigma N^4} \approx \frac{a}{N\sigma}, \quad N \rightarrow \infty. \end{aligned} \quad (2.55)$$

So, from this analysis, we conclude that the variance of the information potential will decrease inversely proportional to  $N$ , which is a comfortable result for estimation. The asymptotic mean integrated square error (AMISE) of the IP is therefore

$$\begin{aligned} \text{AMISE}(\hat{V}(X)) &= E \left[ \int (\hat{V}(X) - V(X))^2 dx \right] \\ &= \frac{\sigma^4}{2} \int (p''(x))^2 dx + \frac{aN(N - 1)(N - 2) + bN(N - 1)}{\sigma N^4} \end{aligned} \quad (2.56)$$

Notice that the AMISE will tend to zero when the kernel size goes to zero and the number of samples goes to infinity with  $N\sigma \rightarrow \infty$ , that is, the IP is a consistent estimator of the 2-norm of the PDF. Unlike the estimators for the mean and variance, the IP is a biased estimator of the 2-norm of the PDF for finite bandwidth kernels. If we compare closely Eqs. (2.49), (2.55), and (2.56) with the well-known Parzen estimation (Eq. (2.12) and [300]) we see that they are asymptotically the same; the bias is proportional to  $\sigma^2$  and the variance decreases proportionally to  $N\sigma$ . The similarity of Eq. (2.56) with kernel density estimation shows that the body of knowledge in density

estimation is directly applicable to the estimation of IP, or that the IP is essentially a kernel estimator of the 2-norm of the PDF. We also can conclude that the estimators of Eqs. (2.14) and (2.18) for the quadratic Renyi's entropy trade bias with variance; that is Eq. (2.14) has larger bias but smaller variance.

## 2.6 Physical Interpretation of Renyi's Entropy Kernel Estimators

There is a useful physical analogy for the kernel estimator in Renyi's entropy as defining an *information potential field* [86]. This analogy has its roots in the link between Renyi's entropy and the norms of the PDF of the data. Indeed, because the kernels in PDF estimation are positive functions that decay with the distance between samples, one can think that one kernel placed on a sample creates a potential field in the sample space, just as physical particles create a gravity field in space. However, in our case the law of interaction is dictated by the kernel shape. The density of samples is measured by the PDF, therefore the potential field in the space of the samples is an approximation of the PDF shape. In this context, samples can be named *information particles* and they interact in the information potential field of the PDF creating *information forces* [86]. The only difference is that this framework must obey the sum constraint of PDFs (so sums are replaced by averages). We explain these concepts next.

Consider  $\hat{V}_2(X)$  in Eq. (2.14) as the average sum of interactions from each sample  $x_j$  in the sample set; that is,  $\hat{V}_2(X) = 1/N \sum_{j=1}^N \hat{V}_2(x_j)$  where

$$\hat{V}_2(x_j) \stackrel{\Delta}{=} 1/N \sum_{i=1}^N \hat{V}_2(x_j; x_i) \quad \text{and} \quad \hat{V}_2(x_j; x_i) = G_{\sigma\sqrt{2}}(x_j - x_i) \quad (2.57)$$

which basically measures the effect of the potential field in the space location occupied by the sample  $x_j$  due to all the other samples  $x_i$ . The sample-by-sample interaction Eq. (2.57) is controlled as we can see by the kernel used in the analysis. The analogy with fields is accurate if we think of the “average” field produced by the samples and this is required to establish the link to PDFs which must add to one.  $\hat{V}_2(x_j)$  can be recognized as the value of the PDF estimated at  $x_j$  so  $\hat{V}_2(x)$ , the estimated PDF with kernels for an arbitrary point  $x$  in the space, can be properly called the information potential field. The IP is just the average value of the information potential field of the sample set (hence the name).

For Gaussian kernels, the derivative of the information potential with respect to the position of sample  $x_j$  is easily evaluated as

$$\frac{\partial}{\partial x_j} \hat{V}_2(x_j) = \frac{1}{N} \sum_{i=1}^N G'_{\sigma\sqrt{2}}(x_j - x_i) = \frac{1}{2N\sigma^2} \sum_{i=1}^N G_{\sigma\sqrt{2}}(x_j - x_i)(x_i - x_j). \quad (2.58)$$

This expression estimates the information force exerted on sample  $x_j$  due to all the other samples. Note that the derivative of the Gaussian evaluated at zero is zero (any kernel that is symmetric, continuous, and maximum at the origin has a similar property). We can also regard Eq. (2.58) as the average contribution of derivatives due to all other samples. Denoting the contribution of a single sample  $x_i$  as  $\hat{F}_2(x_j; x_i)$ , and the overall derivative with respect to  $x_j$  as  $\hat{F}_2(x_j)$ , we get

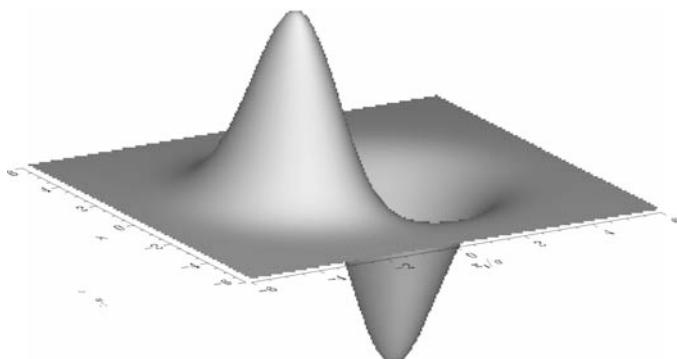
$$\begin{aligned}\hat{F}_2(x_j; x_i) &\stackrel{\Delta}{=} G'_{\sigma\sqrt{2}}(x_j - x_i) \\ \hat{F}_2(x_j) &\stackrel{\Delta}{=} \frac{\partial}{\partial x_j} \hat{V}_2(x_j) = \frac{1}{N} \sum_{i=1}^N \hat{F}_2(x_j; x_i).\end{aligned}\quad (2.59)$$

We name these two quantities the *information force on sample  $x_j$  due to sample  $x_i$*  and the *(total) information force acting on sample  $x_j$* , respectively. Figure 2.4 shows one projection of the information force created by one sample at the origin (Gaussian kernel) in 2D space.

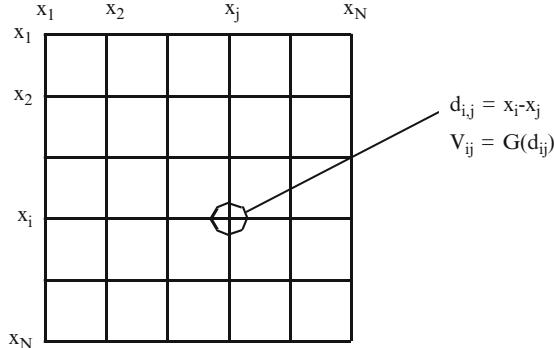
It is instructive to visualize the procedure for the calculation of the information potential and the information force with a Gaussian kernel. For a dataset  $\{x_i\}$  in  $R^n$  two matrices can be defined as

$$\begin{cases} \mathbf{D} = \{\mathbf{d}_{ij}\}, \quad \mathbf{d}_{ij} = \mathbf{x}_i - \mathbf{x}_j \\ \boldsymbol{\varsigma} = \{\hat{V}_{ij}\} \quad \hat{V}_{ij} = G_{\sigma\sqrt{2}}(\mathbf{d}_{ij}), \end{cases}\quad (2.60)$$

where  $\mathbf{D}$  is a matrix of distances, with vector elements in  $R^n$ , and  $\boldsymbol{\varsigma}$  a matrix of scalar values where each element quantifies the interaction between two points in the lattice by the kernel, which gives rise to a similarity matrix. From these quantities, all the quantities of information potential field  $V(i)$  at location  $x_i$ , information force field  $\hat{F}(i)$ , and the information potential  $V(X)$



**Fig. 2.4.** The information force created by one information particle placed at the origin in 2D space (Gaussian kernel) in normalized coordinates  $(x/\sigma, y/\sigma)$  (from [252]).



**Fig. 2.5.** The structure of the matrices  $D$  and  $\zeta$ .

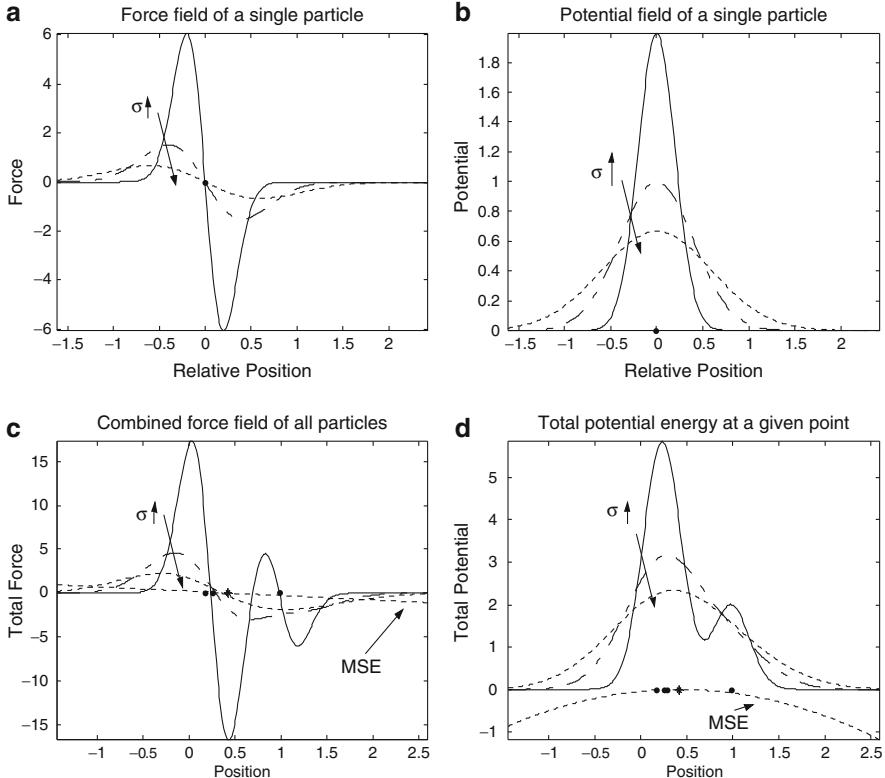
can be easily computed for any Parzen kernel. In fact, for the specific case of the Gaussian kernel they are:

$$\text{fields} \begin{cases} \hat{V}(i) = \frac{1}{N} \sum_{j=1}^N \hat{V}_{i,j} \\ \hat{F}(i) = \frac{-1}{2N\sigma^2} \sum_{j=1}^N \hat{V}_{i,j} d_{i,j} \\ \hat{V}(X) = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \hat{V}_{i,j} = \frac{1}{N} \sum_{i=1}^N \hat{V}(i). \end{cases} \quad (2.61)$$

For further reference, notice that information fields are computed with a single sum over columns (or rows) whereas the information potential requires double sums. Because the computation is done with pairs of samples, it can be visualized in a grid where the sample is the axes that work as pointers to the pairwise distances  $d_{i,j}$ , and the Gaussian is computed directly with this information (Figure 2.5). We can also conclude that the computation complexity of this class of algorithms is  $O(N^2)$  or  $O(N)$  depending on the quantity of interest, where  $N$  is the number of data samples.

### Illustration of Information Forces

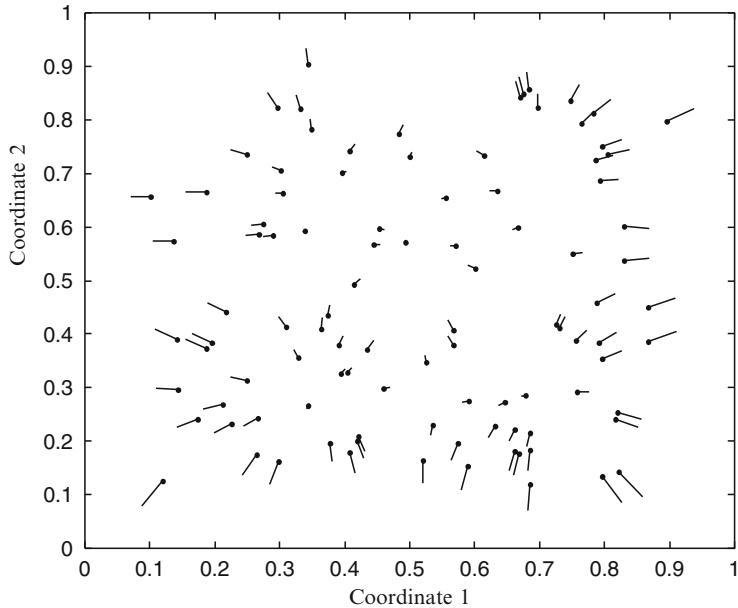
Two numerical examples illustrate the information forces and information potentials in single-dimensional and two-dimensional cases. In the first illustration, we consider the single-dimensional case with the kernel function chosen to be a Gaussian. In Figure 2.6, the one-dimensional information forces and information potential fields are shown for various kernel sizes [86]. The attractive force field of an individual particle centered at the origin is plotted in Figure 2.6a. The forces can be made repulsive by introducing a negative sign in the definition. This procedure corresponds to choosing between minimizing or maximizing the sample entropy. Figure 2.6b shows the information potential at any point due to the existence of this particle at the origin as



**Fig. 2.6.** Forces and potentials as a function of position for different values of kernel size (a) force due to a single particle; (b) potential due to a single particle; (c) overall quadratic force at a given position due to all particles; (d) total quadratic potential at a given position (from [91]).

a function of distance to the origin. To further investigate the effect of additional samples on the potential and force fields, we position three additional randomly located samples. The overall quadratic information force field obtained by superposition of the individual forces of these four particles is shown in Figure 2.6c, and the overall quadratic information potential at a given location is presented as a function of position in Figure 2.6d. All plots include illustrations for various values of the selected kernel size. Notice that, as a consequence of the equivalence with sample variance showed in Property 2.4, as the kernel size increases, the effective force becomes a linear function of distance, and is shown with the label *MSE* in Figure 2.6d. For different kernel functions, different force field definitions can be obtained, changing the adaptation dynamics.

As a second illustration, a snapshot of a two-dimensional entropy maximization scenario is depicted in Figure 2.7, where the particles are bounded to within a unit square and interact under the quadratic force definition with



**Fig. 2.7.** A snapshot of the locations of the information particles and the instantaneous quadratic information forces acting on them to maximize the joint entropy in the two-dimensional unit square (from [91]).

a Gaussian kernel. The objective is to maximize the entropy of the sample ensemble, therefore the forces become repulsive and they stabilize in an arrangement that fills the space uniformly with samples. Given a set of randomly spaced samples in the unit square, when the forces acting on each sample are evaluated, it becomes evident that the information particles are pushed by the other particles in order to move along the direction of maximal entropy. Notice also that the forces tend to be larger for samples away from the center of the cluster (the lines attached to each sample are vectors that display intensity and point to the direction of change).

### Wave Function Interpretation of the Information Potential Estimator

There is another “quantum theory” interpretation of kernel density estimation that is worth presenting [154]. As we have seen, the kernel estimator creates a probability density over the space of the samples. The stationary (time-independent) Schrödinger equation for a particle in the presence of a potential field can be written as

$$\frac{\hbar^2}{2m} \nabla^2 \psi(x) + \psi(x)[E - V_Q(x)] = 0, \quad (2.62)$$

where  $h$  is the Plank's constant,  $m$  the mass of the particle, and the wave function  $\psi$  determines the spatial probability of the particle with  $p(x) = |\psi(x)|^2$ .  $V_Q(x)$  is the “quantum” potential energy as a function of position,  $E$  corresponds to the allowable energy state of the particle, and  $\psi$  becomes the corresponding eigenvector. For the set of information particles with the Gaussian kernel, the wavefunction for a set of  $N$ , one-dimensional, information particles can be written

$$\psi(x) = \sqrt{\frac{1}{N} \sum_{i=1}^N G_\sigma(x - x_i)}.$$

To simplify the derivation and if we are not interested in the physical meaning of the eigenfunctions, we can redefine  $\psi(w)$  as

$$\psi(x) = \sum_{i=1}^N G_\sigma(x - x_i). \quad (2.63)$$

We can also rescale  $V_Q(x)$  such that there is a single free parameter  $\sigma$  in Eq. (2.62) to yield

$$-\frac{\sigma^2}{2} \nabla^2 \psi(x) + V_Q(x) \psi(x) = E \psi(x). \quad (2.64)$$

Solving for  $V_Q(x)$  we obtain

$$V_Q(x) = E + \frac{\sigma/2\nabla^2\psi(x)}{\psi(x)} = E - \frac{1}{2} + \frac{1}{2\sigma^2\psi(x)} \sum_i (x - x_i)^2 e^{-(x-x_i)^2/2\sigma^2}. \quad (2.65)$$

To determine the value of  $V_Q(x)$  uniquely we can require that  $\min V_Q(x) = 0$ , which makes

$$E = -\min \frac{\sigma/2\nabla^2\psi(x)}{\psi(x)}$$

and  $0 \leq E \leq 1/2$ . Note that  $\psi(x)$  is the eigenfunction of  $H$  and  $E$  is the lowest eigenvalue of the operator, which corresponds to the ground state. Given the data set, we expect  $V_Q(x)$  to increase quadratically outside the data region and to exhibit local minima associated with the locations of highest sample density (clusters). This can be interpreted as clustering because the potential function attracts the data distribution function  $\psi(x)$  to its minima, whereas the Laplacian drives it away, producing a complicated potential function in the space. We should remark that in this framework  $E$  sets the scale at which the minima are observed. This derivation can be easily extended to multidimensional data.

We can see that  $V_Q(x)$  in Eq. (2.65) is also a “quantum” potential function that differs from  $V(x)$  in Eq. (2.57) because it is associated with a quantum

description of the information potential. For Gaussian kernels the two fields are similar to each other within the regions with samples and because the derivative of the Gaussian is a Gaussian, but Eq. (2.65) may present advantages because of the intrinsic normalization produced by the eigenvalue that may simplify the search scale for minima of the potential field. We can also estimate the information quantum forces as presented above in Eq. (2.58).

## 2.7 Extension to $\alpha$ -Information Potential with Arbitrary Kernels

Recall the definition of Renyi's  $\alpha$  entropy given in Eq. (2.18). Thus, its non-parametric estimator with arbitrary kernel  $\kappa_\sigma(x)$  with bandwidth  $\sigma$  is given by

$$\hat{V}_\alpha(X) = \frac{1}{N^\alpha} \sum_{j=1}^N \left( \sum_{i=1}^N \kappa_\sigma(x_j - x_i) \right)^{\alpha-1}, \quad (2.66)$$

which can be written as a sum of contributions from each sample  $x_j$ , denoted  $\hat{V}_\alpha(x_j)$ ,

$$\begin{aligned} \hat{V}_\alpha(x_j) &\triangleq \frac{1}{N^{\alpha-1}} \left( \sum_{i=1}^N \kappa_\sigma(x_j - x_i) \right)^{\alpha-1} \\ \hat{V}_\alpha(X) &= \frac{1}{N} \sum_{j=1}^N \hat{V}_\alpha(x_j) \end{aligned} \quad (2.67)$$

for all positive  $\alpha \neq 1$ . Note that this  $\alpha$  information potential can be written as a function of  $\hat{V}_2(x_j)$  as

$$\hat{V}_\alpha(x_j) = \frac{1}{N^{\alpha-2}} \left( \sum_{i=1}^N \kappa_\sigma(x_j - x_i) \right)^{\alpha-2} \frac{1}{N} \sum_{i=1}^N \kappa_\sigma(x_j - x_i) = \hat{p}^{\alpha-2}(x_j) \hat{V}_2(x_j), \quad (2.68)$$

which means that  $\text{IP}_\alpha$  for all integer alpha can be derived conceptually from the quadratic information potential by scaling them by the estimated ( $\alpha$ -2) PDF at the point. Naturally (in analogy with physical potentials), we determine the  $\alpha$  information forces by simply taking the derivative of these information potentials with respect to the particle location (sample value).

$$\begin{aligned} \hat{F}_\alpha(x_j) &\triangleq \frac{\partial}{\partial x_j} \hat{V}_\alpha(x_j) = \frac{\alpha-1}{N^{\alpha-1}} \left( \sum_{i=1}^N \kappa_\sigma(x_j - x_i) \right)^{\alpha-2} \left( \sum_{i=1}^N \kappa'_\sigma(x_j - x_i) \right) \\ &= (\alpha-1) \hat{p}_X^{\alpha-2}(x_j) \hat{F}_2(x_j). \end{aligned} \quad (2.69)$$

This formula defines the total information force acting on sample  $x_j$ , where the quadratic information force is similar to Eq. (2.59), with the exception

that the kernel function need not be specifically Gaussian. In Eq. (2.68), the quadratic force is defined as

$$\hat{F}_2(x_j) \triangleq \frac{1}{N} \left( \sum_{i=1}^N \kappa'_\sigma(x_j - x_i) \right). \quad (2.70)$$

From Eq. (2.69), which is the total information force acting on sample  $x_j$ , and using the additivity of quadratic forces in Eq. (2.70), we can write out the individual contributions of every other sample as

$$\hat{F}_\alpha(x_j; x_i) = (\alpha - 1) \hat{p}_X^{\alpha-2}(x_j) \hat{F}_2(x_j; x_i), \quad (2.71)$$

where we defined

$$\hat{F}_2(x_j; x_i) \triangleq \kappa'_\sigma(x_j - x_i). \quad (2.72)$$

Although we considered above only the single-dimensional case, extensions of these information potential and information force definitions to multidimensional situations is trivial. Note that, in choosing multidimensional kernel functions, some restrictions apply as mentioned in Section 2.3.

Notice that the generalized information forces introduce a scaling factor that depends on the estimated probability density of the corresponding sample and the selected entropy order. Specifically, the baseline is obtained for  $\alpha = 2$ ; that is, the quadratic information potential treats equally the contributions of all the samples. For  $\alpha > 2$ , the scale factor (power of the estimated PDF) in Eq. (2.69) becomes a monotonically increasing function of the PDF value, meaning that compared to the quadratic case, the forces experienced by samples in dense regions of the sample space are amplified. For  $\alpha < 2$ , on the other hand, the opposite takes place, and the forces on sparse regions of the data space are amplified.

This scenario also shows the difficulty of estimating Shannon entropy directly from samples with kernel estimators. The information potential field estimated by Eq. (2.67) becomes constant over the space of the samples for  $\alpha = 1$ , therefore from Eq. (2.69) the force becomes zero. This does not mean that for Shannon's entropy the individual interactions of samples are constant and their forces are zero everywhere in the space, but simply that Eqs. (2.67) and (2.69) that capture macroscopic behavior cannot be applied for  $\alpha = 1$ . Renyi's entropy is discontinuous at  $\alpha = 1$ , therefore the direct substitution of this value in the expressions should be avoided. However, we can use the above estimator formalism for values of alpha close to 1, but we can expect very slow convergence.

## 2.8 Renyi's Divergence and Mutual Information

The structure of probability spaces is much more complex than linear spaces, therefore computing distances in such spaces is nontrivial. The most widely used disimilarity measure is the Kullback-Leibler divergence [188] due to its

nice properties (invariant to reparameterization, monotonicity for Markov chains, and linked locally to the Fisher information matrix that quantifies the Riemannian metric of the space) as we have briefly outlined in Chapter 1. In this section we present Renyi's definition of divergence and mutual information and will also propose two alternate dissimilarity measures in probability spaces that can be easily estimated nonparametrically with the information potential.

### Renyi's $\alpha$ Divergence

Alfred Renyi, in his studies of information theory [264], proposed what is now called the *Renyi's divergence*, intrinsically linked with his definition of entropy and an extension to the KL divergence. The definition of this divergence measure and some of its basic properties are reviewed herein.

Renyi's order- $\alpha$  divergence of  $g(x)$  from  $f(x)$  is defined as [264]

$$D_\alpha(f \parallel g) \triangleq \frac{1}{\alpha-1} \log \int_{-\infty}^{\infty} f(x) \left( \frac{f(x)}{g(x)} \right)^{\alpha-1} dx. \quad (2.73)$$

**Property 2.10.** Renyi's divergence measure has the following properties

- i.  $D_\alpha(f \parallel g) \geq 0, \forall f, g, \alpha > 0.$
- ii.  $D_\alpha(f \parallel g) = 0$  iff  $f(x) = g(x) \forall x \in R.$
- iii.  $\lim_{\alpha \rightarrow 1} D_\alpha(f \parallel g) = D_{KL}(f \parallel g).$

*Proof.* We do the proof of each part separately.

- i. Using Jensen's inequality on the argument of the logarithm in Eq. (2.73), we get

$$\int_{-\infty}^{\infty} f(x) \left( \frac{g(x)}{f(x)} \right)^{1-\alpha} dx \stackrel[0 < \alpha < 1]{\alpha > 1}{\leq} \left( \int_{-\infty}^{\infty} f(x) \left( \frac{g(x)}{f(x)} \right) dx \right)^{1-\alpha} = 1. \quad (2.74)$$

Substituting this result in Eq. (2.73), the desired inequality for all  $\alpha > 0$  is obtained.

- ii. Clearly, if  $g(x) = f(x)$ , then  $D_\alpha(f \parallel g) = 0$ . For the reverse direction, suppose we are given that  $D_\alpha(f \parallel g) = 0$ . Assume  $g(x) \neq f(x)$ , so that  $g(x) = f(x) + \delta(x)$ , where  $\int_{-\infty}^{\infty} \delta(x) dx = 0$ , and  $\exists x \in R$  such that  $\delta(x) \neq 0$ . Consider the divergence between these two PDFs. Equating this divergence to zero, we obtain

$$\begin{aligned} D_\alpha(f \parallel g) &= \frac{1}{1-\alpha} \log \int_{-\infty}^{\infty} f(x) \left( \frac{f(x) + \delta(x)}{f(x)} \right)^{1-\alpha} dx \\ &= \frac{1}{1-\alpha} \log \int_{-\infty}^{\infty} f(x) \left( 1 + \frac{\delta(x)}{f(x)} \right)^{1-\alpha} dx = 0 \quad (2.75) \end{aligned}$$

which implies that

$$\int_{-\infty}^{\infty} f(x) \left( 1 + \frac{\delta(x)}{f(x)} \right)^{1-\alpha} dx = 1 \Rightarrow \left( 1 + \frac{\delta(x)}{f(x)} \right) = 1, \quad \forall x \in R. \quad (2.76)$$

From this last result, we get that  $\delta(x) = 0$ ,  $\forall x \in R$ , which contradicts our initial assumption, therefore, we conclude that  $g(x) = f(x)$ .

iii. Consider the limit of Eq. (2.73) as  $\alpha \rightarrow 1$ .

$$\begin{aligned} \lim_{\alpha \rightarrow 1} D_\alpha(f \parallel g) &= \lim_{\alpha \rightarrow 1} \frac{1}{\alpha - 1} \log \int_{-\infty}^{\infty} f(x) \left( \frac{f(x)}{g(x)} \right)^{\alpha-1} dx \\ &= \frac{\lim_{\alpha \rightarrow 1} \int_{-\infty}^{\infty} -f(x) \left( \frac{f(x)}{g(x)} \right)^{\alpha-1} \log \left( \frac{g(x)}{f(x)} \right) dx}{\lim_{\alpha \rightarrow 1} \int_{-\infty}^{\infty} \left( \frac{f(x)}{g(x)} \right)^{\alpha-1} dx} \\ &= \int_{-\infty}^{\infty} f(x) \log \left( \frac{f(x)}{g(x)} \right) dx = D_{KL}(f \parallel g). \quad (2.77) \end{aligned}$$

Following the same ideas used in deriving the estimator for Renyi's entropy, we can determine a kernel-based resubstitution estimate of Renyi's order- $\alpha$  divergence using Eq. (2.18). Suppose we have the i.i.d. samples  $\{x_g(1), \dots, x_g(N)\}$  and  $\{x_f(1), \dots, x_f(N)\}$  drawn from  $g(x)$  and  $f(x)$ , respectively. The nonparametric estimator for Renyi's divergence obtained with this approach is given as

$$\begin{aligned} D_\alpha(f \parallel g) &= \frac{1}{\alpha - 1} \log E_p \left[ \left( \frac{f(x)}{g(x)} \right)^{\alpha-1} \right] \approx \frac{1}{\alpha - 1} \log \frac{1}{N} \sum_{j=1}^N \left( \frac{\hat{f}(x(j))}{\hat{g}(x(j))} \right)^{\alpha-1} \\ &= \frac{1}{\alpha - 1} \log \frac{1}{N} \sum_{j=1}^N \left( \frac{\sum_{i=1}^N \kappa_\sigma(x_f(j) - x_f(i))}{\sum_{i=1}^M \kappa_\sigma(x_g(j) - x_g(i))} \right)^{\alpha-1} = \hat{D}_\alpha(f \parallel g), \quad (2.78) \end{aligned}$$

with the computational complexity  $O(N^2)$ , the same as the entropy estimator.

Although Renyi's  $\alpha$ -divergence has many of the same properties as KL divergence, it is not as general. In fact, for Shannon's relative entropy, the total information gained by observing a random event  $A$  with probability  $f(x)$  that changes to  $g(x)$  by observing a second event  $B$  can be computed either by averaging the partial gains of information, or by averaging the increase in uncertainty with a negative sign [65], as we have seen in Section 2.5. Shannon information gain for continuous variables

$$I(f \parallel g) = \int f(x) \log \frac{f(x)}{g(x)} dx \quad (2.79)$$

is obtained by averaging over  $f(x)$  the partial gains of information  $\log(f(x)/g(x))$ . Notice that Eq. (2.79) is equivalent to Shannon's relative entropy (KL divergence). However, if Renyi's entropy with  $\alpha \neq 1$  is used to evaluate this gain of information or the negated increase in uncertainty the results differ. Renyi's gain of information by partial increase in information of order  $\alpha$  is [266]

$$\bar{I}_\alpha(f \parallel g) = \frac{1}{1-\alpha} \log \int \frac{f(x)^{2-\alpha}}{g(x)^{1-\alpha}} dx. \quad (2.80)$$

If one uses the measure of uncertainty of order  $\alpha$ , we get Renyi's gain of information of order- $\alpha$  or Renyi's divergence of Eq. (2.73) which is a different quantity (in fact  $\bar{I}_\alpha(f \parallel g) = D_{2-\alpha}(g \parallel f)$ ). Therefore, Shannon relative entropy is the only one for which the sum of average gain of information is equal to the negated average increase of uncertainty. This different behavior between Shannon and Renyi stems from the generalized additivity used in Renyi's definition which excludes the case  $I(f_\Gamma) + I(f_{-\Gamma}) = 0$ , unless  $I_k$  are all the same (uniform distribution), where  $-\Gamma$  is the set of the negated amount of information (i.e.,  $-I_k$ ). This has implications for the definition of Renyi's mutual information as well.

### Renyi's $\alpha$ Mutual Information

Recall that Shannon's mutual information between the components of an  $n$ -dimensional random vector  $X$  is equal to the KL divergence of the joint distribution of  $X$  from the product of the marginal distributions of the components of  $X$  [266]. Similarly, Renyi's  $\alpha$  mutual information is defined as Renyi's divergence between the same quantities. Letting  $p_X(\cdot)$  be the joint distribution and  $p_{X_o}(\cdot)$  be the marginal density of the  $o$ th component, Renyi's mutual information becomes [266]

$$I_\alpha(X) \triangleq \frac{1}{\alpha-1} \log \int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} \frac{p_X^\alpha(x_1, \dots, x_n)}{\prod_{o=1}^n p_{X_o}^{\alpha-1}(x_o)} dx_1 \dots dx_n. \quad (2.81)$$

Once again, it is possible to write a kernel-based estimator for Renyi's mutual information by approximating the joint expectation with the sample mean estimator

$$\begin{aligned} I_\alpha(X) &\triangleq \frac{1}{\alpha-1} \log E_X \left[ \left( \frac{p_X(x_1, \dots, x_n)}{\prod_{o=1}^n p_{X_o}(x_o)} \right)^{\alpha-1} \right] \\ &\approx \frac{1}{\alpha-1} \log \frac{1}{N} \sum_{j=1}^N \left( \frac{p_X(x(j))}{\prod_{o=1}^n p_{X_o}(x_o(j))} \right)^{\alpha-1} \end{aligned} \quad (2.82)$$

and when replacing the PDFs with their Parzen estimators that use consistent kernels between the marginal and joint PDF estimates as mentioned in Property 2.7, the nonparametric mutual information estimator becomes

$$\begin{aligned} \hat{I}_\alpha(X) &\triangleq \frac{1}{\alpha-1} \log \frac{1}{N} \sum_{j=1}^N \left( \frac{\left( \frac{1}{N} \sum_{i=1}^N \kappa_\Sigma(x(j) - x(i)) \right)}{\prod_{o=1}^n \left( \frac{1}{N} \sum_{i=1}^N \kappa_{\sigma_o}(x_o(j) - x_o(i)) \right)} \right)^{\alpha-1} \\ &= \frac{1}{\alpha-1} \log \frac{1}{N} \sum_{j=1}^N \left( \frac{\left( \frac{1}{N} \sum_{i=1}^N \prod_{o=1}^n \kappa_{\sigma_o}(x(j) - x(i)) \right)}{\prod_{o=1}^n \left( \frac{1}{N} \sum_{i=1}^N \kappa_{\sigma_o}(x_o(j) - x_o(i)) \right)} \right)^{\alpha-1} \end{aligned} \quad (2.83)$$

The limit of Eq. (2.83) when  $\alpha \rightarrow 1$  is an estimate of Shannon's mutual information between the random variables under consideration. Therefore, this nonparametric mutual information estimator can be used to estimate directly Renyi's mutual information  $I_\alpha(X)$  from data for  $\alpha$  close to one, but it does not have all the nice properties of Shannon mutual information. Although it is nonnegative and symmetric, it may yield a value greater than 1 (i.e., the information on  $x_1$  given by  $x_2$  can be larger than the information of  $x_1$ , which is a shortcoming). There are many other alternatives to define Renyi's mutual information and unfortunately all of them have shortcomings. See Renyi [266] for a full treatment.

## 2.9 Quadratic Divergences and Mutual Information

As pointed out by Kapur [177], there is no reason to restrict ourselves to Shannon's measure of entropy or to Kullback-Leibler's measure for cross-entropy (density dissimilarity). Entropy and relative entropy are too deep and too complex concepts to be measured by a single measure under all conditions. This section defines divergence and mutual information measures involving only a simple quadratic form of PDFs to take direct advantage of the IP and its nice estimator. A geometric approach is used here.

Looking at the Euclidean space, the two most common families of distance are the sums of difference squares in coordinates and the inner-product

distances, and they are the starting point to derive the corresponding divergences in probability spaces. Equal neighborhoods in the simplex are transformed into spheres of equal size to preserve the Fisher information matrix [32]. This gives rise to a unit sphere where each transformed PMF has coordinates  $\sqrt{p_k}$  (the simplex is transformed to the positive hyperoctant of the sphere). The geodesic distance  $D_G$  between two PMFs  $f$  and  $g$  in the sphere (i.e., the length of the great circle) can be estimated by the cosine of the angle between them, or  $\cos D_G = \sum_k \sqrt{f_k} \sqrt{g_k}$ . This result is related to the argument of the Bhattacharyya distance [39], which is defined (for continuous PDFs) as  $D_B(f, g)$

$$D_B(f, g) = -\ln \left( \int \sqrt{f(x)g(x)} dx \right). \quad (2.84)$$

$D_B(f, g)$  vanishes iff  $f = g$  almost everywhere. We can further establish a link of Eq. (2.84) with Renyi's divergence with  $\alpha = 1/2$  (apart from a scalar). The Chernoff distance [56] or generalized Bhattacharya distance is a non-symmetric measure defined by

$$D_C(f, g) = -\ln \left( \int (f(x))^{1-s} (g(x))^s dx \right) \quad 0 < s < 1 \quad (2.85)$$

which for  $s = 1/2$  yields the Bhattacharyya, and again, apart from the scaling, corresponds to Renyi's divergence for  $\alpha = 1 - s$ .

Instead of the inner product distance we can also measure the distance between  $f$  and  $g$  in a linear projection space of the hyperoctant (chordal distance) as  $D_H = (\sum_k (\sqrt{f_k} - \sqrt{g_k})^2)^{1/2}$ . This result yields the Hellinger's distance [19] which is defined (for continuous densities) as

$$D_H(f, g) = \left[ \int \left( \sqrt{f(x)} - \sqrt{g(x)} \right)^2 dx \right]^{1/2} = \left[ 2 \left( 1 - \int \sqrt{f(x)g(x)} dx \right) \right]^{1/2}. \quad (2.86)$$

Compared with the KL and Renyi's divergences, Hellinger's distance has the advantage of being a difference of PDFs so it avoids stability problems when the denominator PDF is zero. It is also related to the Havrda-Charvat ( $\alpha = 1/2$ ) divergence that is intimately related to  $\alpha$  Renyi's divergence.

After all, Bhattacharyya Eq. (2.84) and Hellinger's distances Eq. (2.86) are angular and Euclidean distances in the hypersphere and their relationship with Renyi's  $\alpha = 1/2$  divergence inspired us to seek definitions of divergences that could benefit from the  $\alpha$  information potential estimator (specifically  $\alpha = 2$ ).

The similarity between two PDFs using the 2-norm is a simple and straightforward distance measure; it obeys all the properties of a distance (including symmetry and the triangular inequality), and can be written as  $D_{ED}(f, g) = \int \sqrt{(f(x) - g(x))^2} dx$ . For simplicity we do not include the square root in the definition because our goal is to use these measures as cost functions, so the Euclidean distance between PDFs is redefined as

$$D_{ED}(f, g) = \int (f(x) - g(x))^2 dx = \int f^2(x) dx - 2 \int f(x)g(x) dx + \int g^2(x) dx \quad (2.87)$$

$D_{ED}(f, g)$  can be recognized as belonging to the same family as the Herlinger distance (chordal distances) but with a different  $\alpha$ -norm. Although being a distance,  $D_{ED}(f, g)$  is sometimes lumped with the divergence terminology in PDF spaces.

The squared distance between the joint PDF and the factorized marginal PDF is called the *quadratic mutual information Euclidean distance* (QMI<sub>ED</sub>), and is written as

$$I_{ED}(X_1, X_2) = D_{ED}(f_{X_1 X_2}(x_1, x_2), f_{X_1}(x_1)f_{X_2}(x_2)). \quad (2.88)$$

$D_{ED}(f, g) \geq 0$  with equality if and only if  $f(x) = g(x)$  almost everywhere and the integrals involved are all quadratic forms of PDFs. Obviously, the QMI<sub>ED</sub> between  $X_1$  and  $X_2$  is nonnegative and is zero if and only if  $f_{X_1 X_2}(x_1, x_2) = f_{X_1}(x_1)f_{X_2}(x_2)$ ; that is,  $X_1$  and  $X_2$  are independent random variables. There is no strict theoretical justification that the QMI<sub>ED</sub> is an appropriate measure for dependence between two variables. However, it can be shown that  $D_{ED}(f, g)$  is a lower bound for the KL divergence [319], therefore when one maximizes  $D_{ED}(f, g)$ , we are also maximizing KL. For multiple variables, the extension of QMI<sub>ED</sub> interpreted as a multivariate dissimilarity measure is straightforward:

$$I_{ED}(X_1, \dots, X_k) = D_{ED}(f_X(x_1, \dots, x_k), \prod_{i=1}^k f_{X_i}(x_i)),$$

where  $f_X(x_1, \dots, x_k)$  is the joint PDF, and  $f_{X_i}(x_i), (i = 1, \dots, k)$  are marginal PDFs.

The other possible PDF divergence is related to the Battacharyya distance. Formally it can be derived from the Cauchy-Schwarz inequality [276]:

$$\sqrt{\int f^2(x) dx \int g^2(x) dx} \geq \int f(x)g(x) dx, \quad (2.89)$$

where equality holds if and only if  $f(x) = cg(x)$  for a constant scalar  $c$ . If  $f(x)$  and  $g(x)$  are PDFs (i.e.,  $\int f(x) dx = 1$  and  $\int g(x) dx = 1$ ), then  $f(x) = cg(x)$  implies  $c = 1$ . So, for two PDFs  $f(x)$  and  $g(x)$  equality holds if and only if  $f(x) = g(x)$ . Similarly to  $D_{ED}(f, g)$  for the estimators we normally use the square of Eq. (2.89) to simplify the calculations. Thus, we may define the Cauchy-Schwarz divergence for two PDFs as

$$D_{CS}(f, g) = -\log \frac{\left( \int f(x)g(x) dx \right)^2}{\int f^2(x) dx \int g^2(x) dx}, \quad (2.90)$$

$D_{CS}(f, g) \geq 0$ , where the equality holds if and only if  $f(x) = g(x)$  almost everywhere and the integrals involved are all quadratic forms of PDFs.  $D_{CS}(f, g)$  is symmetric but it does not obey the triangular inequality.

Let us look closely at Eq. (2.87). One can immediately recognize the first and last terms as the quadratic information potential of  $f(x)$  and  $g(x)$ , respectively. The middle term  $\int f(x)g(x)dx$  is called the *cross information potential (CIP)*, and basically estimates the interactions on locations in the space dictated by the dataset  $f(x)$  in the potential created by the dataset  $g(x)$  (or viceversa). This is really the term that measures the “distance” between the two PDFs, because the other two are simply normalizing terms. The  $D_{CS}(f, g)$  of Eq. (2.90) can be rewritten as

$$D_{CS}(f, g) = \log \int f(x)^2 dx + \log \int g(x)^2 dx - 2 \log \int f(x)g(x)dx, \quad (2.91)$$

where all the three terms of  $D_{ED}(f, g)$  appear also in  $D_{CS}(f, g)$ , simply with a logarithmic weighting. Based on  $D_{CS}(f, g)$ , we define the *Cauchy–Schwarz quadratic mutual information* ( $QMI_{CS}$ ) between two variables  $X_1$  and  $X_2$  as

$$I_{CS}(X_1, X_2) = D_{CS}(f_X(x_1, x_2), f_{X_1}(x_1)f_{X_2}(x_2)), \quad (2.92)$$

where the notations are the same as above. Directly from above,  $I_{CS}(X_1, X_2) \geq 0$  meets the equality if and only if  $X_1$  and  $X_2$  are independent random variables. So,  $I_{CS}$  is an appropriate measure of independence. This measure is a geodesic distance in the sphere, therefore the Cauchy–Schwarz divergence may also be appropriate as a dependence measure in cases where the PDFs exist in the sphere. For multivariate variables, the extension of  $QMI_{CS}$  is also straightforward:

$$I_{CS}(X_1, \dots, X_k) = D_{CS}(f_X(x_1, \dots, x_k), \prod_{i=1}^k f_{X_i}(x_i)).$$

### Cauchy–Schwarz Divergence and Renyi’s Relative Entropy

Recently, Lutwak, et al [205] defined a new Renyi’s divergence called the relative  $\alpha$ -Renyi entropy between  $f(x)$  and  $g(x)$  as

$$D_{R_\alpha}(f, g) = \log \frac{\left(\int_R g^{\alpha-1}(x)f(x)\right)^{\frac{1}{(1-\alpha)}} \left(\int_R g^\alpha(x)\right)^{1/\alpha}}{\left(\int_R f^\alpha(x)\right)^{\frac{1}{\alpha(1-\alpha)}}}. \quad (2.93)$$

Note that the denominator in the argument of the log now contains an integral that is more robust than Renyi’s original definition of Eq. (2.73). So  $f(x)$  could be zero at some points of the domain but overall the integral is well defined, thus avoiding numerical issues of Eq. (2.73). Again, for  $\alpha \rightarrow 1$ , this gives  $D_{KL}(f || g)$ . In particular, for  $\alpha = 2$  Eq. (2.93) is exactly the Cauchy–Schwarz divergence of Eq. (2.90). This is a very interesting relation because it

provides an information-theoretic interpretation both for the Cauchy–Schwarz divergence and also for the integral of the product of PDFs. Indeed, we can rewrite the Cauchy–Schwarz divergence in terms of Renyi's quadratic entropy as

$$\begin{aligned} D_{CS}(X, Y) &= -\log \left( \int f(x)g(x) \right)^2 dx + \log \left( \int f(x)^2 dx \right) \\ &\quad + \log \left( \int g(x)^2 dx \right) \\ &= 2H_2(X; Y) - H_2(X) - H_2(Y), \end{aligned} \tag{2.94}$$

where the first term can be shown to be the *quadratic Renyi's cross-entropy* [259] (and should not be confused with the joint entropy of  $X$  and  $Y$ ). The similarity of this expression with Shannon's mutual information in Eq. (1.10) is striking if we think in terms of cross-entropy versus joint entropy.

## 2.10 Information Potentials and Forces in the Joint Space

The interactions among the samples interpreted as information particles for the case of divergence are substantially more involved than IP because of the different information potential fields that exist. In essence one has to realize that each probability density function creates its own information potential field, and that particle interactions are produced by weighted sums of each potential field computed in the joint space [340]. We illustrate the principles for the calculation of the Euclidean distance and QMIED.

### Euclidean and Cauchy–Schwarz Divergence Estimators

The divergences are composed of three different information potential fields, each specified by the location of the samples from  $f(x)$ , from  $g(x)$ , and the cross-information potential field. Because the potentials are additive, we can compute one at a time and add the result as specified by Eq. (2.86). The information potential estimated by the Gaussian kernel for each PDF is given by

$$\begin{aligned} \hat{V}_f &= \frac{1}{N^2} \sum_{i=1}^N \sum_{i=j}^N G_{\sqrt{2}\sigma}(x_f(i) - x_f(j))^2 \\ \hat{V}_g &= \frac{1}{N^2} \sum_{i=1}^N \sum_{i=j}^N G_{\sqrt{2}\sigma}(x_g(i) - x_g(j))^2 \\ \hat{V}_c &= \frac{1}{N^2} \sum_{i=1}^N \sum_{i=j}^N G_{\sqrt{2}\sigma}(x_f(i) - x_g(j))^2, \end{aligned} \tag{2.95}$$

where for simplicity we assume that we have the same number of samples ( $N$ ) in each dataset and the same kernel size  $\sigma$ .  $V_C$  is the cross-information potential estimator and basically measures the interaction of the field created by  $f(x)$  on the locations specified by  $g(x)$ . The Euclidean and Cauchy–Schwarz information potential fields are therefore:

$$\begin{aligned}\hat{D}_{ED}(f, g) &= \hat{V}_{ED} = \hat{V}_f + \hat{V}_g - 2\hat{V}_c \\ \hat{D}_{CS}(f, g) &= \hat{V}_{CS} = \log \frac{\hat{V}_f \hat{V}_g}{\hat{V}_c^2}.\end{aligned}\quad (2.96)$$

The computation of the Euclidean and Cauchy–Schwarz information forces exerted on each sample  $x_i$  can be easily achieved using the additive rule of derivatives

$$\begin{aligned}\frac{\partial \hat{V}_{ED}}{\partial x_i} &= \frac{\partial \hat{V}_f}{\partial x_i} + \frac{\partial \hat{V}_g}{\partial x_i} - 2 \frac{\partial \hat{V}_c}{\partial x_i} \\ \frac{\partial \hat{V}_{CS}}{\partial x_i} &= \frac{1}{\hat{V}_f} \frac{\partial \hat{V}_f}{\partial x_i} + \frac{1}{\hat{V}_g} \frac{\partial \hat{V}_g}{\partial x_i} - \frac{2}{\hat{V}_c} \frac{\partial \hat{V}_c}{\partial x_i}.\end{aligned}\quad (2.97)$$

The computation of the Euclidean and Cauchy–Schwarz divergences can proceed in a fashion very similar to the information potential defining a matrix of distances and of scalars as given by Eq. (2.61). See [340] for a full treatment.

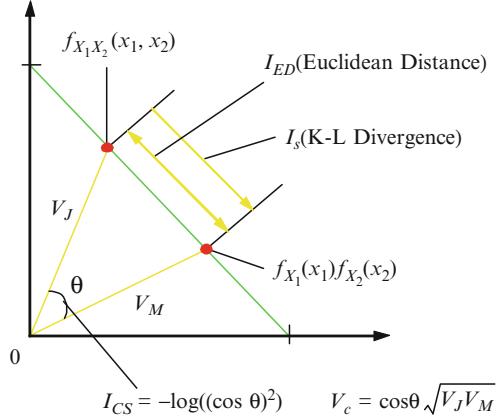
### Generalized Information Potential (GIP) for Quadratic Mutual Information

Both QMI<sub>ED</sub> and QMI<sub>CS</sub> can be written in a very similar way to  $D_{ED}(f, g)$  and  $D_{CS}(f, g)$  but they are a little more complex due to the fact that we have two variables and the existence of the joint and the product of the marginals. We can decompose the overall expressions Eqs. (2.88) and (2.92) in the following three terms,

$$\begin{cases} V_J = \iint f_{X_1 X_2}(x_1, x_2)^2 dx_1 dx_2 \\ V_M = \iint (f_{X_1}(x_1) f_{X_2}(x_2))^2 dx_1 dx_2 \\ V_c = \iint f_{X_1 X_2}(x_1, x_2) f_{X_1}(x_1) f_{X_2}(x_2) dx_1 dx_2 \end{cases}\quad (2.98)$$

where  $V_J$  is the IP of the joint PDF,  $V_M$  is the IP of the factorized marginal PDF, and  $V_C$  is the generalized cross information potential. Just like for the quadratic divergences, this is really the term that measures the interactions between the two information potentials, whereas the other two are proper normalizations. With these three terms, both QMIs yield

$$\begin{cases} I_{ED} = V_J - 2V_c + V_M \\ I_{CS} = \log V_J - 2 \log V_c + \log V_M \end{cases}\quad (2.99)$$



**Fig. 2.8.** Geometrical interpretation of quadratic mutual information.

Figure 2.8 shows the illustration of the geometrical interpretation of all these quantities in the 2D simplex (for the case of discrete random variables).  $I_S$ , as previously mentioned, is the KL divergence between the joint PDF and the factorized marginal PDF,  $I_{ED}$  is the squared Euclidean distance between these two PDFs, and  $I_{CS}$  is related to the angle between these two PDFs.

For the estimation of each of the potentials in Eq. (2.98) the following notation is used: subscripts denote the input components, and indices represent sums over samples. For a given dataset  $\{\mathbf{x}(i) = (x_1(i), x_2(i))^T | i = 1, \dots, N\}$  of a two-dimensional variable  $X = (x_1, x_2)^T$ , the joint and marginal PDFs define a joint ( $V_J$ ), a marginal ( $V_M$ ) and a cross ( $V_C$ ) information potential field from Eq. (2.98). Using Gaussian kernels to estimate the joint and the marginals yields,

$$\left\{ \begin{array}{l} \hat{f}_{X_1 X_2}(x_1, x_2) = \frac{1}{N} \sum_{i=1}^N G_\sigma(\mathbf{x} - \mathbf{x}(i)) \\ \hat{f}_{X_1}(x_1) = \frac{1}{N} \sum_{i=1}^N G_\sigma(x_1 - x_1(i)) \\ \hat{f}_{X_2}(x_2) = \frac{1}{N} \sum_{i=1}^N G_\sigma(x_2 - x_2(i)). \end{array} \right. \quad (2.100)$$

Because information potential fields are additive, we can estimate independently the three terms in  $QMI_{ED}$  or  $QMI_{CS}$  of Eq. (2.99) based only on the given dataset.

Note that  $V_J$ , which exists over the joint space, can be decomposed for radially symmetric kernels in a product of interactions along each of the variables  $G(\mathbf{x}_i - \mathbf{x}_j) = G(x_{1i} - x_{1j})G(x_{2i} - x_{2j})$ , where  $\mathbf{x} = (x_1, x_2)^T$ . The generalized cross-information potential  $V_C$  of Eq. (2.98) is the potential that seems more difficult to compute, therefore it is illustrated here. Starting from the definition of the information potential, we obtain

$$\begin{aligned}
\hat{V}_C &= \iint \hat{f}(x_1, x_2) \hat{f}(x_1) \hat{f}(x_2) dx_1 dx_2 \\
&= \iint \left[ \frac{1}{N} \sum_{k=1}^N G_\sigma(x_1 - x_1(k)) G_\sigma(x_2 - x_2(k)) \right] \left[ \frac{1}{N} \sum_{i=1}^N G_\sigma(x_1 - x_1(i)) \right] \\
&\quad \times \left[ \frac{1}{N} \sum_{j=1}^N G_\sigma(x_2 - x_2(j)) \right] dx_1 dx_2 \\
&= \frac{1}{N} \sum_{i=1}^N \frac{1}{N} \sum_{j=1}^N \frac{1}{N} \sum_{k=1}^N \int G_\sigma(x_1 - x_1(i)) G_\sigma(x_1 - x_1(k)) dx_1 \\
&\quad \times \int G_\sigma(x_2 - x_2(k)) G_\sigma(x_2 - x_2(j)) dx_2 \\
&= \frac{1}{N} \sum_{k=1}^N \left[ \frac{1}{N} \sum_{i=1}^N G_{\sqrt{2}\sigma}(x_1(k) - x_1(i)) \right] \left[ \frac{1}{N} \sum_{j=1}^N G_{\sqrt{2}\sigma}(x_2(k) - x_2(j)) \right]
\end{aligned} \tag{2.101}$$

Notice that the GCIP for QMI requires  $O(N^3)$  computation.  $V_M$  can be further factorized as two marginal information potentials  $V_1$  and  $V_2$

$$\begin{aligned}
\hat{V}_M &= \iint \hat{f}_{X_1}^2(x_1) \hat{f}_{X_2}^2(x_2) dx_1 dx_2 \\
\hat{V}_1 &= \iint \hat{f}_{X_1}^2(x_1) dx_1 \\
\hat{V}_2 &= \iint \hat{f}_{X_2}^2(x_2) dx_2
\end{aligned} \tag{2.102}$$

Therefore the final expressions are

$$\begin{aligned}
\hat{V}_k(i, j) &= G_{\sqrt{2}\sigma}(x_k(i) - x_k(j)), \\
\hat{V}_k(i) &= \frac{1}{N} \sum_{j=1}^N \hat{V}_k(i, j), \hat{V}_k = \frac{1}{N} \sum_{i=1}^N \hat{V}_k(i), k = 1, 2 \\
\hat{V}_J &= \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \hat{V}_1(i, j) \hat{V}_2(i, j) \\
\hat{V}_M &= \hat{V}_1 \hat{V}_2 \text{ with } \hat{V}_k = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \hat{V}_k(i, j), k = 1, 2 \\
\hat{V}_C &= \frac{1}{N} \sum_{i=1}^N \hat{V}_1(i) \hat{V}_2(i).
\end{aligned} \tag{2.103}$$

So, the estimated Euclidean Mutual information ( $\text{QMI}_{\text{ED}}$ ) and the estimated Cauchy-Schwarz Mutual Information ( $\text{QMI}_{\text{CS}}$ ) are given by

$$\begin{aligned}\hat{I}_{\text{ED}}(X_1, X_2) &= \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \hat{V}_1(i, j) \hat{V}_2(i, j) + \hat{V}_1 \hat{V}_2 - \frac{1}{N} \sum_{i=1}^N \hat{V}_1(i) \hat{V}_2(i) \\ \hat{I}_{\text{CS}}(X_1, X_2) &= \log \frac{\left( \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \hat{V}_1(i, j) \hat{V}_2(i, j) \right) (\hat{V}_1 \hat{V}_2)}{\left( \frac{1}{N} \sum_{i=1}^N \hat{V}_1(i) \hat{V}_2(i) \right)^2}.\end{aligned}\quad (2.104)$$

From the above, we can see that both QMIs can be expressed as interactions between the marginal information potential fields at different levels:  $V_1(i, j)V_2(i, j)$  is the level of the sample-to-sample interactions from each marginal (the joint field),  $V_1(i)V_2(j)$  is the level of one full marginal field acting on a single sample (the GCIP), and  $V_1V_2$  is the interaction between both marginal potential fields (product of marginals).  $\hat{I}_{\text{ED}}$  is called the Euclidean generalized information potential ( $\text{GIP}_{\text{ED}}$ ), and  $\hat{I}_{\text{CS}}$  is the Cauchy-Schwartz generalized information potential ( $\text{GIP}_{\text{CS}}$ ).

The quadratic mutual information and the corresponding cross information potential can be easily extended to the case with multiple variables (e.g.,  $X = (x_1, \dots, x_k)^T$ ). In this case, we have similar IPs and marginal IPs as in Eq. (2.104). Then we have the  $\text{QMI}_{\text{ED}}$  and  $\text{QMI}_{\text{CS}}$  and their corresponding  $\text{GIP}_{\text{ED}}$  and  $\text{GIP}_{\text{CS}}$  as follows,

$$\begin{aligned}\hat{I}_{\text{ED}}(X_1, \dots, X_K) &= \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \prod_{k=1}^K \hat{V}_k(i, j) - \frac{2}{N} \sum_{i=1}^N \prod_{k=1}^K \hat{V}_k(i) + \prod_{k=1}^K \hat{V}_k \\ \hat{I}_{\text{CS}}(X_1, \dots, X_K) &= \log \frac{\left( \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \prod_{k=1}^K \hat{V}_k(i, j) \right) \prod_{k=1}^K \hat{V}_k}{\left( \frac{1}{N} \sum_{i=1}^N \prod_{k=1}^K \hat{V}_k(i) \right)^2}.\end{aligned}\quad (2.105)$$

## Generalized Information Forces

Three different potentials contribute to the generalized information potential, but because the derivative is distributive with respect to addition, one can still operate on each term independently. The cases of  $\text{GIP}_{\text{ED}}$  and  $\text{GIP}_{\text{CS}}$  are slightly different because of the logarithm, but the procedure is to take the derivative of Eq. (2.104) with respect to a given sample, yielding

$$\begin{aligned}\hat{F}_{\text{ED}}(i) &= \frac{\partial \hat{I}_{\text{CS}}}{\partial x_k(i)} = \frac{\partial \hat{V}_j}{\partial x_k(i)} - \frac{2\partial \hat{V}_C}{\partial x_k(i)} + \frac{\partial \hat{V}_k}{\partial x_k(i)} \\ \hat{F}_{\text{CS}}(i) &= \frac{\partial \hat{I}_{\text{CS}}}{\partial x_k(i)} = \frac{1}{V_j} \frac{\partial \hat{V}_j}{\partial x_k(i)} - \frac{2}{V_C} \frac{\partial \hat{V}_C}{\partial x_k(i)} + \frac{1}{V_k} \frac{\partial \hat{V}_k}{\partial x_k(i)}.\end{aligned}\quad (2.106)$$

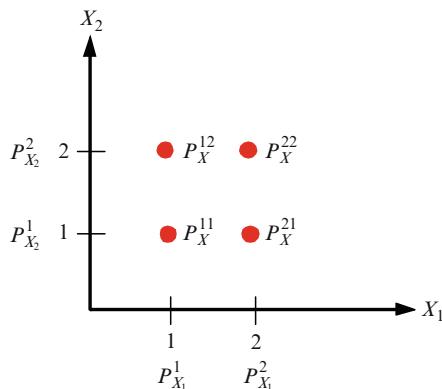
The case for multiple variables can be readily obtained in a similar way (see [340] for a full treatment).

### A simple example

To understand the similarities and differences among the dissimilarity measures  $I_S$ ,  $I_{ED}$ , and  $I_{CS}$ , let's look at a simple case with two discrete random variables  $X_1$  and  $X_2$  as shown in Figure 2.9. It is trivial to apply these definitions to discrete events. We exemplify here the QMI<sub>ED</sub> and QMI<sub>CS</sub>. For the discrete variables  $X_1$  and  $X_2$  with probability distribution  $\{P_{X_1}(i); i = 1, \dots, n\}$  and  $\{P_{X_2}(j); j = 1, \dots, m\}$ , respectively, and the joint probability distribution  $\{P_X(i, j); i = 1, \dots, n; j = 1, \dots, m\}$ , the QMI<sub>ED</sub> and QMI<sub>CS</sub> are

$$\begin{cases} I_{ED}(X_1, X_2) = \sum_{i=1}^n \sum_{j=1}^m (P_X(i, j) - P_{X_1}(i)P_{X_2}(j))^2 \\ I_{CS}(X_1, X_2) = \log \frac{\left( \sum_{i=1}^n \sum_{j=1}^m (P_X(i, j))^2 \right) \left( \sum_{i=1}^n \sum_{j=1}^m (P_{X_1}(i)P_{X_2}(j))^2 \right)}{\sum_{i=1}^n \sum_{j=1}^m (P_X(i, j)P_{X_1}(i)P_{X_2}(j))^2} \end{cases}. \quad (2.107)$$

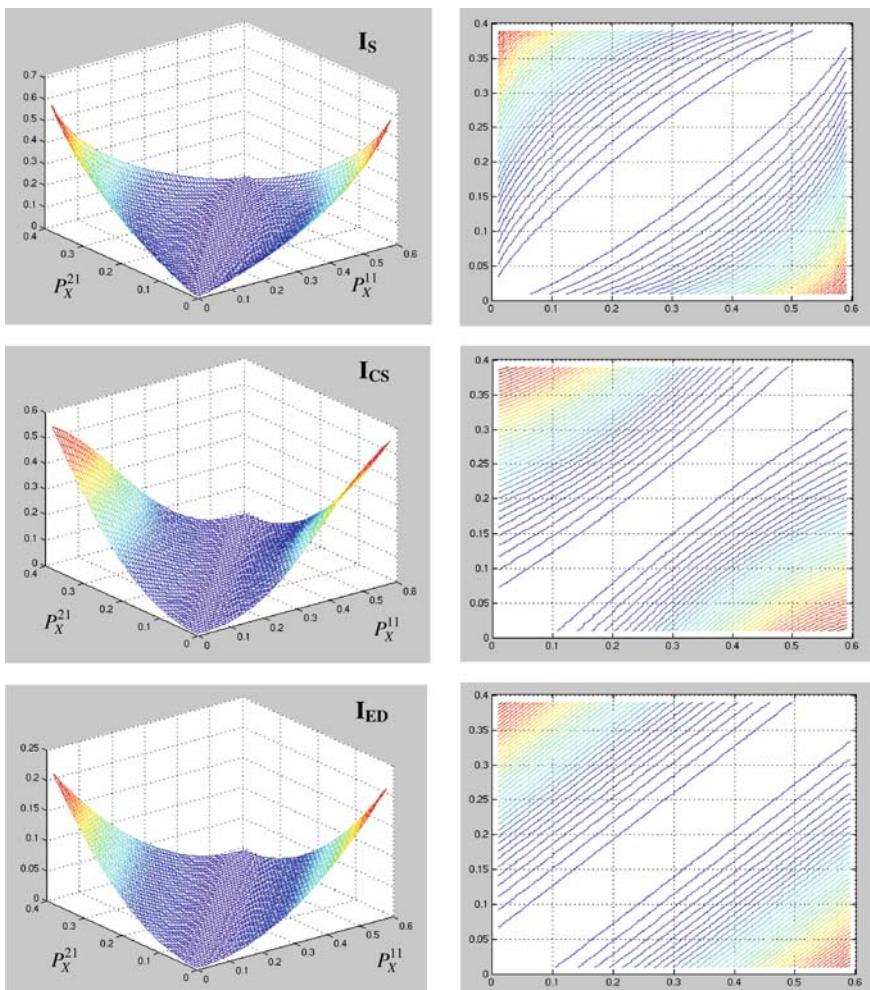
$X_1$  can take the values 1 or 2 with a probability  $P_{X_1} = (P_{X_1}(1), P_{X_1}(2))$ ; that is,  $P(X_1 = 1) = P_{X_1}(1)$  and  $P(X_1 = 2) = P_{X_1}(2)$ . Similarly  $X_2$  can take the values 1 or 2 with the probability  $P_{X_2} = (P_{X_2}(1), P_{X_2}(2))$  where  $P(X_2 = 1) = P_{X_2}(1)$  and  $P(X_2 = 2) = P_{X_2}(2)$ . The joint probability distribution is  $P_X = (P_X(1, 1), P_X(1, 2), P_X(2, 1), P_X(2, 2))$ ; where  $P_X(1, 1) = P((X_1, X_2) = (1, 1))$  and likewise for the other cases. Obviously,



**Fig. 2.9.** The 2D data for the example.

$P_{X_1}(1) = P_X(1, 1) + P_X(1, 2)$ ,  $P_{X_1}(2) = P_X(2, 1) + P_X(2, 2)$ ,  $P_{X_2}(1) = P_X(1, 1) + P_X(2, 1)$ , and  $P_{X_2}(2) = P_X(1, 2) + P_X(2, 2)$ . In the following figures related to this example, the probability variables are simplified as  $P_{X_1}^1 = P_X(1, 1)$ ,  $P_X^{11} = P_X(1, 1)$ , etc.

First, let's look at the case with the marginal distribution of  $X_1$  fixed as  $P_{X_1} = (0.6, 0.4)$ . Then the free parameters left are  $P_X(1, 1)$  from  $[0, 0.6]$  and  $P_X(2, 1)$  from  $[0, 0.4]$ . When  $P_X(1, 1)$  and  $P_X(2, 1)$  change in these ranges, the values of  $I_S$ ,  $I_{ED}$ , and  $I_{CS}$  can be easily calculated. The right graphs in Figure 2.10 show the contour plots of the corresponding left surfaces (contour means that each line has the same functional value).



**Fig. 2.10.** The surfaces and contours of  $I_S$ ,  $I_{ED}$ , and  $I_{CS}$  versus  $P_X(1, 1)$  and  $P_X(2, 1)$

These graphs show that although the contours of the three measures are different, they reach the minimum value 0 in the same line  $P_X(1,1) = 1.5P_X(2,1)$  where the joint probabilities equal the corresponding factorized marginal probabilities. And the maximum values, although different, are also reached at the same points  $(P_X(1,1), P_X(2,1)) = (0.6, 0)$  and  $(0, 0.4)$  where the joint probabilities are

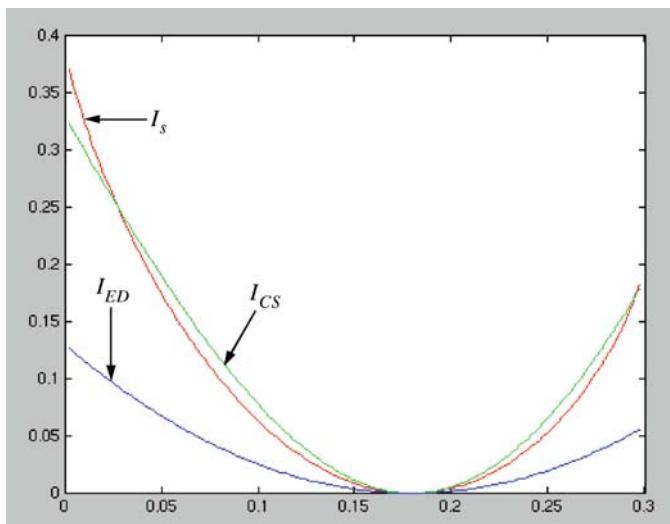
$$\begin{bmatrix} P_X(1,2) & P_X(2,2) \\ P_X(1,1) & P_X(2,1) \end{bmatrix} = \begin{bmatrix} 0 & 0.4 \\ 0.6 & 0 \end{bmatrix} \text{ and } \begin{bmatrix} P_X(1,2) & P_X(2,2) \\ P_X(1,1) & P_X(2,1) \end{bmatrix} = \begin{bmatrix} 0.6 & 0 \\ 0 & 0.4 \end{bmatrix},$$

respectively.

If the marginal probability of  $X_2$  is further fixed (e.g.  $P_{X2} = (0.3, 0.7)$ ), then the free parameter is  $P_X(1,1)$  from 0 to 0.3, which can be regarded as the previous setting with a further constraint specified by  $P_X(1,1) + P_X(2,1) = 0.3$ . In this case, both marginal probabilities of  $X_1$  and  $X_2$  are fixed, the factorized marginal probability distribution is also fixed and only the joint probability distribution will change. Figure 2.11 shows how the three measures change with  $P_X(1,1)$ , from which we can see that the minima are reached at the same point  $P_X(1,1) = 0.18$ , and the maxima are also reached at the same point  $P_X(1,1) = 0$ ; that is,

$$\begin{bmatrix} P_X(1,2) & P_X(2,2) \\ P_X(1,1) & P_X(2,1) \end{bmatrix} = \begin{bmatrix} 0.6 & 0.1 \\ 0 & 0.3 \end{bmatrix}.$$

From this simple example, we can see that although the three measures are different, they have the same minimum point and also have the same



**Fig. 2.11.**  $I_S$ ,  $I_{ED}$ , and  $I_{CS}$  versus  $P_X(1,1)$ .

maximum points in this particular case. It is known that both Shannon's mutual information  $I_S$  and  $\text{QMI}_{\text{ED}}(I_{\text{ED}})$  are convex functions of PDFs, and  $I_{\text{ED}}$  is a lower bound for  $I_S$ . From the above graphs, we can confirm this fact and also reach the conclusion that  $\text{QMI}_{\text{CS}}(I_{\text{CS}})$  is not a strictly convex function of PDFs.

## 2.11 Fast Computation of IP and CIP

One of the practical difficulties of the information potential, cross-information potential and ITL quantities in general, such as  $D_{\text{ED}}$  and  $D_{\text{CS}}$  and  $\text{QMI}_{\text{ED}}$  and  $\text{QMI}_{\text{CS}}$  is that the calculations are  $O(N^2)$  or  $O(N^3)$ , respectively. This section presents an effort to make the estimation of IP faster using two techniques: one based on the fast Gauss transform (FGT) [122] and the other using the incomplete Cholesky decomposition to exploit the Gram matrix band structure that for kernels possess rapidly decreasing eigenvalues, particularly in low dimensions [100].

### Fast Gauss Transform

The fast multipole method is a very interesting and important family of fast evaluation algorithms that have been developed over the past two decades to enable rapid calculation of approximations, with arbitrary accuracy, to large matrix-vector products of the form  $\mathbf{A}\mathbf{d}$  where the elements of  $\mathbf{A}$  are  $a_{i,j} = \sum_i \sum_j \varphi(x_i - x_j)$  with  $\varphi$  a nonlinear fast decaying positive function of the argument [121]. The fast Gauss transform [122] is a special case derived for efficient calculation of weighted sums of unidimensional Gaussians at a point  $y_i$ ,

$$S(y_i) = \sum_{j=1}^N w_j e^{-(y_j - y_i)^2 / 4\sigma^2} \quad i = 1, \dots, M \quad (2.108)$$

The FGT has been applied to many areas including astrophysics, kernel density estimation, and machine learning algorithms decreasing the computation from  $O(NM)$  to  $O(N + M)$  where  $N$  is the number of samples (sources) and  $M$  the number of points where the evaluation is required. The computational savings come from two facts, both related to the shifting property of the Gaussian function

$$\begin{aligned} e^{-\left(\frac{y_j - y_i}{\sigma}\right)^2} &= e^{-\left(\frac{y_j - y_C - (y_i - y_C)}{\sigma}\right)^2} \\ &= e^{-\left(\frac{y_j - y_c}{\sigma}\right)^2} \sum_{n=0}^{\infty} \frac{1}{n!} \left(\frac{y_i - y_c}{\sigma}\right)^n h_n\left(\frac{y_j - y_c}{\sigma}\right), \end{aligned} \quad (2.109)$$

which means that a Gaussian centered at  $y_j$  can be shifted to a sum of Hermite polynomials times a Gaussian, all centered at  $y_c$ . First, the Hermite polynomials  $h_n(y)$  given by

$$h_n(y) = (-1)^n \frac{d^n \exp(-x^2)}{dx^n} \quad (2.110)$$

are very efficient in the approximation and a small order  $p$  is normally sufficient; that is,

$$\exp\left(\frac{-(y_j - y_i)^2}{4\sigma^2}\right) = \sum_{n=0}^{p-1} \frac{1}{n!} \left(\frac{y_i - y_C}{2\sigma}\right)^n h_n\left(\frac{y_j - y_C}{2\sigma}\right) + \varepsilon(p),$$

where  $\varepsilon(p)$  is the error associated with the truncation of the expansion at order  $p$ . The second savings is that there is no need to evaluate every Gaussian at every point. Instead a  $p$ -term sum is computed around a small number  $y_c$  of cluster centers with  $O(Np)$  computation with Eq. (2.109). These sums are then shifted to the  $y_i$  desired locations and computed in another  $O(Mp)$  operation. In practice, an expansion around a single center is not always accurate over the entire domain of interest. A tiling of the space is constructed and the Gaussian function is expanded at multiple centers with the FGT. To efficiently subdivide the space, a very simple greedy algorithm called *furthest-point clustering* [117] can be used, which computes a data partition with a maximum radius at most twice the optimum for the problem. The direct implementation of furthest-point clustering has running time  $O(BN)$ , with  $B$  the number of clusters.

If one recalls the definition of the IP this algorithm can be immediately applied, remembering that now the sources and the locations where the expansion needs to be computed coincide. If we apply this expansion to the IP  $V(y)$ , we obtain

$$V(y) \approx \frac{1}{2\sigma N^2 \sqrt{\pi}} \sum_{j=1}^N \sum_{b=1}^B \sum_{n=0}^{p-1} \frac{1}{n!} h_n\left(\frac{y_j - y_{Cb}}{2\sigma}\right) C_n(b), \quad (2.111)$$

where  $B$  is the number of clusters used with centers  $y_{Cb}$ , and  $C_n(b)$  is defined by

$$C_n(b) = \sum_{y_i \in B} \left(\frac{y_j - y_{Cb}}{2\sigma}\right)^n, \quad (2.112)$$

From the above equation, we can see that the total number of operations required is  $O(BpN)$  per data dimension. The order  $p$  of the expansion depends on the desired accuracy required (normally 4 or 5), and is independent of  $N$ . In addition to the complexity reduction, the other appeal of the FGT is that the code becomes parallelizable due to the clustering step.

### Taylor Series for Multiple Dimensions

The extension to more than one dimension of the previous algorithm is done by treating the multivariate Gaussian as a Kronecker product of univariate Gaussians. Following the multi-index notation of the original FGT papers, we define the multidimensional Hermite function as

$$h_\alpha(\mathbf{y}) = h_{\alpha_1}(y_1)h_{\alpha_2}(y_2)\cdots h_{\alpha_d}(y_d), \quad (2.113)$$

where  $\mathbf{y} = (y_1, \dots, y_d)^T \in R^d$  and  $\alpha = (\alpha_1, \dots, \alpha_d) \in N^d$ . As can be expected the algorithm scales up very poorly with dimension due to this product form.

An alternative method introduced by Yang et al. [345] is to expand the Gaussian function into a multivariate Taylor series. The Gaussian function is factorized as

$$\begin{aligned} \exp\left(-\frac{\|\mathbf{y}_j - \mathbf{y}_i\|^2}{4\sigma^2}\right) &= \exp\left(-\frac{\|\mathbf{y}_j - c\|^2}{4\sigma^2}\right) \\ &\times \exp\left(-\frac{\|\mathbf{y}_i - c\|^2}{4\sigma^2}\right) \exp\left(2\frac{(\mathbf{y}_j - c) \cdot (\mathbf{y}_i - c)}{4\sigma^2}\right). \end{aligned} \quad (2.114)$$

In the third term of (2.114), the product of the evaluation at two different points (called the entanglement) is split by expanding the exponential into a Taylor series as

$$\exp\left(2\frac{(\mathbf{y}_j - c) \cdot (\mathbf{y}_i - c)}{4\sigma^2}\right) = \sum_{\alpha \geq 0} \frac{2^{|\alpha|}}{\alpha!} \left(\frac{\mathbf{y}_j - c}{2\sigma}\right)^\alpha \left(\frac{\mathbf{y}_i - c}{2\sigma}\right)^\alpha + \varepsilon(\alpha), \quad (2.115)$$

where the factorial and the length of  $\alpha$  are defined, respectively, as  $\alpha! = \alpha_1! \alpha_2! \cdots \alpha_d!$  and  $|\alpha| = \alpha_1 + \alpha_2 + \cdots + \alpha_d$ . The IP can then be written using this form as

$$V_T(\mathbf{y}) \approx \frac{1}{N^2 (4\pi\sigma^2)^{d/2}} \sum_{j=1}^N \sum_B \sum_{\alpha \geq 0} C_\alpha(B) \exp\left(-\frac{\|\mathbf{y}_j - c_B\|^2}{4\sigma^2}\right) \left(\frac{\mathbf{y}_j - c_B}{2\sigma}\right)^\alpha, \quad (2.116)$$

where the coefficients  $C_\alpha$  are given by

$$C_\alpha(B) = \frac{2^{|\alpha|}}{\alpha!} \left\{ \sum_{e_i \in B} \exp\left(-\frac{\|\mathbf{y}_i - c_B\|^2}{4\sigma^2}\right) \left(\frac{\mathbf{y}_i - c_B}{2\sigma}\right)^\alpha \right\}. \quad (2.117)$$

The coefficients  $C_\alpha$  are lexicographically ordered before storage because the expansion of multivariate polynomials can be performed efficiently in this form. For a  $d$ -dimensional polynomial of order  $p$ , all terms are stored in a vector of length

$$r_{p,d} = \binom{p+d}{d} = \frac{(p+d)!}{d!p!}.$$

If the series is truncated at order  $p$ , then the number of terms is  $r_{p,d}$  which is much less than  $\Pi_p^d = p^d$  in higher dimensions. The total computational complexity is  $O(BNr_{p,d})$ , where  $B$  is the number of clusters.

These algorithms decrease the number of computations appreciably when estimating entropy and divergence in ITL with the Gaussian kernel because

the computations become  $O(N)$ . However, we have to remember that they are not exact evaluations, therefore the number of terms in the expansions and the number of clusters have to be determined appropriately according to the application. Coprocessors for desktop computers in the GigaFlop range have been developed for astrophysics applications [179], but they lack the flexibility required for ITL where the number of dimensions of the problem, the kernel type are free parameters and where the computations are much more general than just evaluating forces.

### Incomplete Cholesky Decomposition

Any  $N \times N$  symmetric positive definite matrix  $\mathbf{K}$  can be expressed as  $\mathbf{K} = \mathbf{G}^T \mathbf{G}$  where  $\mathbf{G}$  is an  $N \times N$  lower triangular matrix with positive diagonal entries. This decomposition is known as the Cholesky decomposition which is a special case of the LU decomposition for a symmetric positive definite matrix [116]. However, if the eigenvalues of  $\mathbf{K}$  drop rapidly, then the matrix can be approximated by a  $N \times D$  ( $D \leq N$ ) lower triangular matrix  $\tilde{\mathbf{G}}$  with arbitrary accuracy; that is,  $\|\mathbf{K} - \tilde{\mathbf{G}}^T \tilde{\mathbf{G}}\| < \varepsilon$  where  $\varepsilon$  is a small positive number of choice and  $\|\cdot\|$  is a suitable matrix norm. This decomposition is called the incomplete Cholesky decomposition (ICD) [116]. It is observed that in kernel learning [100], depending on the eigenstructure of the matrix, even  $D \ll N$  provides desired accuracy in practice. Although computation involving  $\mathbf{K}$  can be largely simplified using  $\mathbf{G}$ , computing  $\tilde{\mathbf{G}}$  itself appears as an overhead, but fortunately there are efficient algorithms to accomplish this task [116]. The particular algorithm in the following table takes a greedy approach and tries to minimize the trace of the residual  $\mathbf{K} - \tilde{\mathbf{G}}^T \tilde{\mathbf{G}}$ . Its space complexity is  $O(ND)$  and the time complexity is  $O(ND^2)$ , exactly the same complexity as the factorization of  $\tilde{\mathbf{G}}$ . We provide the algorithm below.

### Fast Computation of IP

The information potential can be written in terms of a symmetric positive Gram matrix as

$$\hat{V}(X) = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \kappa(x_i - x_j) = \frac{1}{N^2} \mathbf{1}_N^T \mathbf{K}_{XX} \mathbf{1}_N = \frac{1}{N^2} \left\| \mathbf{1}_N^T \tilde{\mathbf{G}}_{XX} \right\|_2^2, \quad (2.118)$$

where  $\mathbf{1}_N$  is a vector of all 1 and size  $N$ . The computation decreases from  $O(N^2)$  to  $O(ND^2)$ , and we have obtained precisions of  $10^{-6}$  for 1000 sample datasets, while reducing the computation time 100-fold [292]. The quadratic mutual information algorithms of Eq. (2.97) use only the IP so they can be easily written as

**Algorithm 1** Incomplete Cholesky decomposition

---

```

1: Input:  $X$ ,  $\kappa$  and  $\epsilon$ , Output:  $\mathbf{G}$ 
2:  $\mathbf{D} \in \mathbb{R}^{n \times 1}$ ,  $\mathbf{P} = [1, 2, \dots, n]^\top$ 
3:  $\mathbf{G}(:, 1) = \mathbf{K}(:, 1)$ 
4: for  $i = 1 : n$  do
5:   if  $i = 1$  then
6:      $\mathbf{D}(i : n) = \text{diag}(\mathbf{K})$ 
7:   else
8:      $\mathbf{D}(i : n) = \text{diag}(\mathbf{K}(i : n, i : n)) -$ 
9:        $(\mathbf{G}(i : n, 1 : i - 1) \circ \mathbf{G}(i : n, 1 : i - 1)) \mathbf{1}_{i-1}$ 
10:  end if
11:  if  $\sum_{j=i}^n \mathbf{D}(j) < \epsilon$  then
12:    BREAK
13:  end if
14:   $j^* = \arg \max_{i \leq j \leq n} \mathbf{D}(j)$ 
15:   $\mathbf{P}(i) \leftrightarrow \mathbf{P}(j^*)$ 
16:   $\mathbf{G}(i, 1 : i - 1) \leftrightarrow \mathbf{G}(j^*, 1 : i - 1)$ 
17:   $\mathbf{G}(i, i) = \sqrt{\mathbf{D}(j^*, j^*)}$ 
18:   $\mathbf{G}(i + 1 : n, i) = (\mathbf{K}(\mathbf{P}(i + 1 : n), \mathbf{P}(i)) -$ 
19:     $\mathbf{G}(i + 1 : n, 1 : i - 1) * (\mathbf{G}(i, 1 : i - 1))^\top) / \mathbf{G}(i, i)$ 
20: end for
21: Sort rows of  $\mathbf{G}$  according to  $\mathbf{P}$ 

```

---

$$\begin{aligned} \hat{I}_{ED} &= \frac{1}{N^2} \mathbf{1}_{\mathbf{D}_x}^T \left( \tilde{\mathbf{G}}_{XX}^T \tilde{\mathbf{G}}_{YY} \circ \tilde{\mathbf{G}}_{XX}^T \tilde{\mathbf{G}}_{YY} \right) \mathbf{1}_{\mathbf{D}_y} \\ &\quad + \frac{1}{N^4} \left\| \mathbf{1}_N^T \tilde{\mathbf{G}}_{XX} \right\|_2^2 \left\| \mathbf{1}_N^T \tilde{\mathbf{G}}_{YY} \right\|_2^2 \\ &\quad - \frac{2}{N^3} \left( \mathbf{1}_N^T \tilde{\mathbf{G}}_{XX} \right) \left( \tilde{\mathbf{G}}_{XX}^T \tilde{\mathbf{G}}_{YY} \right) \left( \tilde{\mathbf{G}}_{YY}^T \mathbf{1}_N \right) \end{aligned} \quad (2.119)$$

$$\begin{aligned} \hat{I}_{CS} &= \log \frac{\mathbf{1}_{\mathbf{D}_x}^T \left( \tilde{\mathbf{G}}_{XX}^T \tilde{\mathbf{G}}_{YY} \circ \tilde{\mathbf{G}}_{XX}^T \tilde{\mathbf{G}}_{YY} \right) \mathbf{1}_{\mathbf{D}_y} \left\| \mathbf{1}_N^T \tilde{\mathbf{G}}_{XX} \right\|_2^2 \left\| \mathbf{1}_N^T \tilde{\mathbf{G}}_{YY} \right\|_2^2}{\left( \left( \mathbf{1}_N^T \tilde{\mathbf{G}}_{XX} \right) \left( \tilde{\mathbf{G}}_{XX}^T \tilde{\mathbf{G}}_{YY} \right) \left( \tilde{\mathbf{G}}_{YY}^T \mathbf{1}_N \right) \right)^2}. \end{aligned} \quad (2.120)$$

In these expressions the symbol  $\circ$  denotes the elementwise matrix multiplication (Hadamard or Schur product). The computational complexity decreases dramatically from  $O(N^3)$  to  $O(N(D_x^2 + D_y^2 + D_x D_y))$ .

### Fast Computation of the CIP

Unfortunately, the cross information potential does not yield a symmetric positive definite Gram matrix, therefore the above algorithm cannot be directly applied. However, one can augment the matrix to make it symmetric: if the

Gram matrix for the CIP is denoted  $\mathbf{K}_{XY}$ , we create a matrix of double size given by

$$\mathbf{K}_{ZZ} = \begin{vmatrix} \mathbf{K}_{XX} & \mathbf{K}_{XY} \\ \mathbf{K}_{XY} & \mathbf{K}_{YY} \end{vmatrix}.$$

This may seem a waste, but it turns out that in many ITL descriptors each one of the parts of this matrix is needed as we show below. The CIP then can be written as

$$\hat{V}(X, Y) = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \kappa(x_i - y_j) = \frac{1}{N^2} \mathbf{e}_1^T \mathbf{K}_{zz} \mathbf{e}_2 = \frac{1}{N^2} (\mathbf{e}_1^T \tilde{\mathbf{G}}_{ZZ}) (\tilde{\mathbf{G}}_{ZZ} \mathbf{e}_2), \quad (2.121)$$

where

$$\mathbf{e}_1 = \underbrace{\{1, \dots, 1\}}_N, \underbrace{\{0, \dots, 0\}}_N^T \text{ and } \mathbf{e}_2 = \underbrace{\{0, \dots, 0\}}_N, \underbrace{\{1, \dots, 1\}}_N^T.$$

The computational complexity of the CIP is also  $O(ND^2)$ . The divergences of Eq. (2.86) and (2.90) that use the CIP can be written in matrix form as

$$\begin{aligned} \hat{D}_{ED} &= \frac{1}{N^2} (\mathbf{e}_1^T \tilde{\mathbf{G}}_{ZZ}) (\tilde{\mathbf{G}}_{ZZ}^T \mathbf{e}_1) + \frac{1}{N^2} (\mathbf{e}_2^T \tilde{\mathbf{G}}_{ZZ}) (\tilde{\mathbf{G}}_{ZZ}^T \mathbf{e}_2) \\ &\quad - \frac{2}{N^2} (\mathbf{e}_1^T \tilde{\mathbf{G}}_{ZZ}) (\tilde{\mathbf{G}}_{ZZ}^T \mathbf{e}_2) \end{aligned} \quad (2.122)$$

$$\hat{D}_{CS} = \log \frac{(\mathbf{e}_1^T \tilde{\mathbf{G}}_{ZZ}) (\tilde{\mathbf{G}}_{ZZ}^T \mathbf{e}_1) (\mathbf{e}_2^T \tilde{\mathbf{G}}_{ZZ}) (\tilde{\mathbf{G}}_{ZZ}^T \mathbf{e}_2)}{\left( (\mathbf{e}_1^T \tilde{\mathbf{G}}_{ZZ}) (\tilde{\mathbf{G}}_{ZZ}^T \mathbf{e}_2) \right)^2}. \quad (2.123)$$

The computational complexity is identical to the CIP. The advantage of the ICD with respect to the FGT is the simpler data structures for the computation, in as much as everything is done in vector matrix products.

## 2.12 Conclusion

This chapter presented the definition of Renyi's family of entropies, their meaning and relationship with Shannon, and their impact in developing nonparametric estimators for entropy. In particular the argument of the log of quadratic Renyi's entropy called here the information potential, can be estimated directly from data with kernels. The IP can be considered on a par with nonparametric estimators of mean and variance, but unlike them it depends upon one free parameter that needs to be estimated from the data structure and controls the bias and variance for finite datasets. This brings flexibility to the designer, but also requires proper selection. The simple Silverman's rule of density estimation is normally sufficient when data are low-dimensional,

but more involved techniques such as cross-validation are required for more accurate results. This dependence on the kernel size makes the IP appropriate primarily for relative comparisons within the same dataset, but when the goal is to optimize a cost function this is perfectly suitable.

From the IP estimator we developed a physical interpretation for the PDF estimated with kernels as a potential field, where samples interact with each other under information forces. From the information potential we proposed two dissimilarity measures in probability spaces that can also be estimated directly from data because they are functions of the IP. One important part of the chapter addresses a detailed treatment of the properties of these estimators in adaptation in as much as this is going to be instrumental for the rest of the chapters, including an analysis of the mean and variance of the IP.

The chapter presents all the necessary equations to implement ITL quantities and they are used extensively throughout the book. The estimator of entropy and ITL divergences are  $O(N^2)$  and the estimator for QMI is  $O(N^3)$ , therefore we also presented two approximations to the computation that provides tremendous speedups in most cases, by using the concept of fast Gauss transform and incomplete Cholesky decomposition.

---

# Adaptive Information Filtering with Error Entropy and Error Correntropy Criteria

Deniz Erdogmus and Weifeng Liu

## 3.1 Introduction

This chapter formulates a new cost function for adaptive filtering based on Renyi's quadratic error entropy. The problem of estimating the linear system parameters  $\mathbf{w} = [w_0, \dots, w_{M-1}]^T$  in the setting of Figure 3.1 where  $x(n)$ , and  $z(n)$  are random variables can be framed as model-based inference, because it relates measured data, uncertainty, and the functional description of the system and its parameters. The desired response  $z(n)$  can be thought of as being created by an unknown transformation of the input vector  $\mathbf{x} = [x(n), \dots, x(n - M + 1)]^T$ . Adaptive filtering theory [143, 284] addresses this problem using the MSE criterion applied to the error signal,  $e(n) = z(n) - f(\mathbf{w}, x(n))$

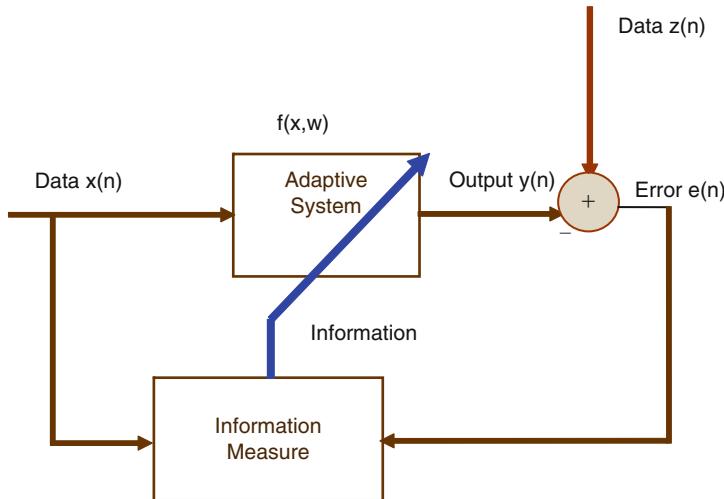
$$J_w(e(n)) = E[(z(n) - f(\mathbf{w}, x(n)))^2] \quad (3.1)$$

when the linear filter is a finite impulse response filter (FIR);

$$y(n) = \sum_{k=0}^{M-1} w_k x(n - k). \quad (3.2)$$

The optimal solution was derived in Chapter 1 (Eqs. (1.26) to (1.34)) and reads  $\mathbf{w} = \mathbf{R}^{-1}\mathbf{p}$ , where  $\mathbf{R}$  is the autocorrelation matrix of the input  $x(n)$  ( $M \times M$ ) and  $\mathbf{p}$  is the cross-correlation vector between the input and desired signals. The gradient search approach yields the famed LMS algorithm,  $w_k(n+1) = w_k(n) + \eta e(n)x_k(n)$ ,  $k = 0, \dots, M-1$ , which approaches the neighborhood of the optimal solution incrementally with only two multiplications per parameter.

The goal of the chapter is to substitute the MSE of Eq. (3.1) with an information measure, more specifically an entropic criterion, and to provide



**Fig. 3.1.** Error entropy adaptation.

some important properties. One of the achievements is that we show that the entropic error shares properties of the M-estimators developed by Huber [157]. We define a novel function called *correntropy* and contrast it to MSE as well as with the entropic cost to provide understanding. Unfortunately, there is no known analytical solution to this class of cost functions because the optimization problem is nonlinear in the weights.

### 3.2 The Error Entropy Criterion (EEC) for Adaptation

It can be argued that MSE is not always the best possible criterion to use in adaptation. In fact, the minimization of MSE is just taking into consideration the second-order moment of the error distribution, which is optimal only for Gaussian distributed errors. In cases where the error distribution is not Gaussian, it makes sense to study alternate cost functions for adaptation. The traditional way to handle this shortcoming is to include higher-order nonlinearity in the errors [4, 78], and the best-known algorithms are perhaps the least mean fourth (LMF) algorithm [328], the  $L_p$  power [244], or the mixed norm algorithms [313].

Here we take a different approach using information-theoretical concepts, and propose the error entropy criterion (EEC) where the goal of adaptation should be to remove as much uncertainty as possible from the error signal. In the ideal case, the error PDF should be a delta function, meaning that all the uncertainty in the error was removed, or equivalently, all the information contained in the input-desired signal pairs was translated into the weights of the system. Combining the ideas of optimization and information theory

presented in Chapter 1, this can be accomplished by calculating the entropy of the error and minimizing it with respect to the free parameters;

$$\min_w H[e] \quad \text{s.t.} \quad e = z - f(\mathbf{x}, \mathbf{w}) \quad \& \quad E[e] = 0, \quad (3.3)$$

which is called the minimization of the error entropy algorithm or MEE for short [87]. Instead of using Shannon's entropy definition we substitute Renyi's quadratic entropy explained in Chapter 2 to take advantage of its estimator provided by the information potential (IP) estimator of Eq. (2.14), and in general for any  $\alpha \neq 1$  with Eq. (2.18). More recently this method was extended to the  $(h, \phi)$  entropies [53]. Recall that Renyi's quadratic entropy of the error is defined as

$$\begin{cases} H_2(e) = -\log V(e) \\ V(e) = E[p(e)]. \end{cases} \quad (3.4)$$

Notice that Renyi's quadratic entropy is a monotonic function of the negative of  $V(e)$  for  $\alpha > 1$ . Therefore, for the purpose of adaptation, the logarithm can be dropped (it will not change the location of the stationary point of the cost function in parameter space), and minimization (or maximization) of entropy will correspond to maximization (minimization) of  $V(e)$ ; that is,

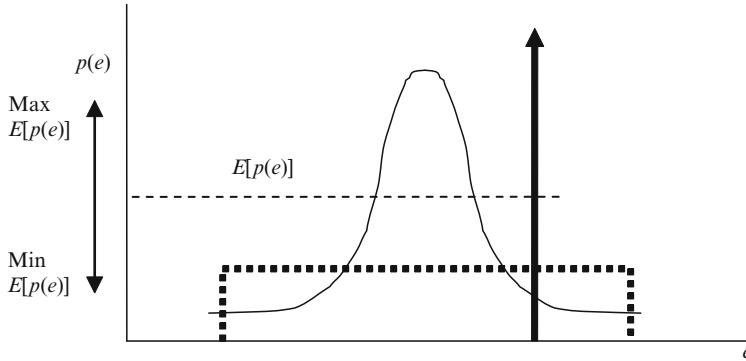
$$\min_w H_2(e) = \max_w V(e). \quad (3.5)$$

This simplifies the explanation and the learning algorithms. Therefore, in analogy with Eq. (1.27), the optimal weights can be found by computing [332],

$$\frac{\partial H_2(e)}{\partial \mathbf{w}} \rightarrow \frac{\partial V(e)}{\partial \mathbf{w}} = 0. \quad (3.6)$$

### 3.3 Understanding the Error Entropy Criterion

Dropping the logarithm is inconceivable in an information-theoretic context therefore, we seek here an explanation for the role of the expected value of the PDF as a cost function for adaptation. Recall from Eq. (3.4) that  $V(e) = E[p(e)]$ , which makes  $V(e)$  a function of both the error and its PDF unlike the MSE cost. Let us define  $\xi = p(e)$  to yield  $V(e) = E[\xi]$ , or in other words, we are nonlinearly transforming the error by its own PDF, which means that we have left the error space and actually are performing the adaptation on a transformed variable  $\xi$ . Figure 3.2 depicts this mapping. Why is this mapping useful? As is well known in statistics, the PDF contains all the relevant statistical structure in the data. We claim that when the goal is to capture in a single number the statistical structure of the error samples through a transformation, the PDF is the natural nonlinear mapping from the data space to the transformed space. Of course through  $\xi$  we are transforming samples into samples, so we still do not have the single number we desire, but if we take the mean value of  $\xi$  we easily achieve our goal and obtain Eq. (3.4).



**Fig. 3.2.** The result of maximizing/minimizing the IP in the space of the error for an arbitrary error PDF.

In adaptation, the error (the horizontal axis of Figure 3.2) is effectively under the control of the parameters of the adaptive system because of the error dependence on  $y$  which is a function of  $w$  (Eq. (3.2)). With Eq. (3.4), changing the parameters of the learning system will also change the shape of the error PDF, or in other words, in EEC not only the error samples but also the set of functions that can be created in the mapping are under the control of the learning system. This is clearly a much more powerful criterion than MSE [78].

The EEC works on the vertical axis, and in the case of Renyi's quadratic entropy is simply  $E[p(e)]$ , a scalar that turns out to be the mean of  $\xi$ . To minimize entropy we want obviously to make the error PDF a delta function anywhere in the space of the error, which implies a maximization of  $E[p(e)]$  as Eq. (3.5) specifies. On the contrary, if we want to maximize entropy we want the PDF to be flat (for finite range) or a Gaussian when the range is infinite, which leads to a minimization of  $E[p(e)]$  in the vertical axis. In essence, the specific values of the error are not important in EEC because the cost is just constraining the mean of its PDF. We gain flexibility and power because a single number now is related to the mean of the error distribution instead of being just a fixed function (the power) of the error as in MSE. As a conclusion, if one thinks about this interpretation of EEC using  $E[p(e)]$ , information theory is really not necessary, although our work was initially motivated by entropy. But, we have to be much more careful when taking the derivative with respect to the weights, because there is a functional dependence involved through  $p(e)$ .

Let us assume a linear system for simplicity,  $e = z - \mathbf{w}^T \mathbf{x}$ . Note that the cost is a function of the probability density function (PDF) of  $e$ , denoted  $p_e(\cdot)$ , which depends on the error, the parameter  $w$ , and in general also on the joint PDF of the input  $X$  and desired response  $Z$ . Further note that when we change the weights, there is a functional dependence among  $e$ ,  $\mathbf{w}$ , and  $p(e)$ , therefore we need to compute the total derivative of  $V(e, \mathbf{w}, p_{xz})$  w.r.t. the

weights, instead of the partial derivative we normally utilize in MSE. From Eq. (3.6) taking the total derivative of  $V$  w.r.t. to the weights yields

$$\frac{d}{dw} V(e, \mathbf{w}, p_{XZ}) = \frac{d}{dw} \int p_e(e, \mathbf{w}, p_{XZ})^2 de \quad (3.7)$$

or

$$\frac{d}{dw} V(e, \mathbf{w}, p_{XZ}) = 2E \left[ \frac{d}{dw} p_e(e, \mathbf{w}, p_{XZ}) \right] \Big|_{e=e}. \quad (3.8)$$

The difficulty is that both the function  $p_e(\cdot)$  and the evaluation point change when weights are being adapted, so we have to compute partial derivatives for each, which yields

$$\begin{aligned} E \left[ \frac{d}{dw} p_e(e, \mathbf{w}, p_{XZ}) \right] \Big|_{e=e} &= E \left[ \frac{\partial p_e(e, \mathbf{w}, p_{XZ})}{\partial e} \frac{\partial e}{\partial \mathbf{w}} + \frac{\partial p_e(e, \mathbf{w}, p_{XZ})}{\partial \mathbf{w}} \right] \Big|_{e=e} \\ &= E \left[ \frac{\partial p_e(e, \mathbf{w}, p_{XZ})}{\partial e} \frac{\partial e}{\partial \mathbf{w}} \Big|_{e=e} + \frac{\partial p_e(e, \mathbf{w}, p_{XZ})}{\partial \mathbf{w}} \right] \end{aligned} \quad (3.9)$$

so

$$\begin{aligned} E \left[ \frac{d}{dw} p_e(e, \mathbf{w}, p_{XZ}) \right] \Big|_{e=e} &= E \left[ (\partial_1 p_e)(e, \mathbf{w}, p_{XZ}) \frac{\partial e}{\partial \mathbf{w}} + (\partial_2 p_e)(e, \mathbf{w}, p_{XZ}) \right] \\ &= E [(\partial_1 p_e)(e, \mathbf{w}, p_{XZ}) X + (\partial_2 p_e)(e, \mathbf{w}, p_{XZ})], \end{aligned} \quad (3.10)$$

where  $\partial_1, \partial_2$  correspond, respectively, to the partial derivative operator on the first and second argument of the function (the partial with respect to the third term is zero, in as much as the joint does not depend on the weights). In simplified notation, one could write

$$\begin{aligned} \frac{d}{dw} V(e, \mathbf{w}, p_{XZ}) &= 2E \left[ \left( \frac{\partial p_e}{\partial e} \right) X + \left( \frac{\partial p_e}{\partial \mathbf{w}} \right) \right] = 0 \\ &\rightarrow E \left[ \left( \frac{\partial p_e}{\partial e} \right) X \right] = -E \left[ \left( \frac{\partial p_e}{\partial \mathbf{w}} \right) \right]. \end{aligned} \quad (3.11)$$

If  $p_{XZ}$  is differentiable, then at extrema of the cost function  $V$ , one of which will be the optima, the gradient is zero:  $\partial V(e)/\partial w = 0$ . If we compare Eq. (3.11) with the MSE in Eq. (1.28) we see two big differences: the first term includes the derivative of the error PDF instead of the derivative of the error square, but apart from this difference, this term quantifies how a change of weight affects the cost through the error. Notice that the input  $X$  appears in both equations as it should because it refers to the propagation of the error through the filter topology. The second term in Eq. (3.11) quantifies how a change in the weight affects the cost by changing the shape of the PDF. This term does not exist in MSE because in Eq. (1.28) there is a direct relation between the error and the weight. It can be shown that when the PDF is the Gaussian Eq. (3.11) yields back Eq. (1.28).

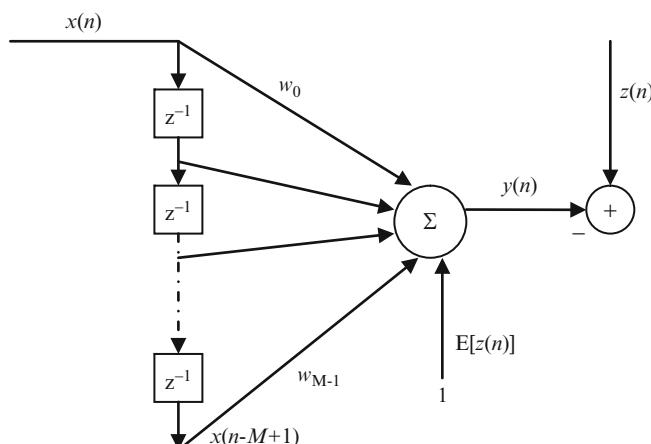
### Utilizing the Error Entropy Criterion in Adaptation

The interpretation of the EEC and how it differs from MSE must be well understood if we are going to use it effectively in adaptation. This is the purpose of this section. It is apparent that EEC does not constrain the mean value of the error, because it is acting on the projected random variable (the vertical axis of Figure 3.2) and only the PDF shape that achieves the cost goal is constrained, immaterial of where it is located in the error space. But practical adaptive problems require that the mean of the error be zero such that the system output approaches the desired response. Therefore, the designer must take care of the error mean by an additional procedure when using the EEC. The simplest approach is to use an extra bias term at the output of the FIR that is set by the mean of the desired response (Figure 3.3). Therefore a linear combiner with bias, that is,  $y(n) = \sum_{k=0}^{M-1} w_k x(n-k) + \bar{z}(n)$ , is always assumed in our discussions.

There are more principled ways of achieving the same goal as explained later, but this solution is simple and works for desired responses with symmetric PDFs.

### Optimizing Error Entropy Criterion with Estimators

The derivation of Eq. (3.11) is very nice, but the issues associated with the implementation of this criterion seem mind boggling: the PDF is not known, its estimators are ill-posed, and furthermore the criterion is changing through iterations because the parameters of the system are changing! The beauty of the ITL methodology is that we can approximate Eq. (3.11) by estimating directly the 2-norm of the PDF with the information potential estimator and working directly with data samples.



**Fig. 3.3.** Compensating the mean indeterminacy of EEC with a bias weight.

The minimization can be achieved by maximizing the information potential estimator  $\hat{V}_2(e)$ . Recall from Chapter 2 that, given a batch of  $N$  samples, the estimator for  $V_2(e)$  with the Gaussian kernel is

$$\hat{V}_2(e) = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N G_{\sqrt{2}\sigma}(e_i - e_j). \quad (3.12)$$

According to the gradient ascent approach (because the interest is in maximizing  $V(e)$ ) and a linear combiner (Eq. (1.32)), the  $k$ th weight of the FIR at time  $n + 1$  can be adapted as

$$w_k(n+1) = w_k(n) + \eta \nabla_k \hat{V}_2(n) \quad k = 0, \dots, M-1, \quad (3.13)$$

where we denote the IP for the past  $N$  samples as  $\hat{V}_2(n)$ . Substituting Eq. (3.12), the gradient for  $w_k$  can be estimated as

$$\nabla_k \hat{V}_2(n) = \frac{\partial \hat{V}_2(n)}{\partial w_k} = \frac{\partial}{\partial w_k} \left( \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N G_{\sqrt{2}\sigma}(e(n-i) - e(n-j)) \right). \quad (3.14)$$

Exchanging the derivative with the double sum and applying the chain rule through the filter structure this can be further written as

$$\nabla_k \hat{V}_2(n) = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \frac{\partial G_{\sqrt{2}\sigma}(e(n-i) - e(n-j))}{\partial(e(n-i) - e(n-j))} \frac{\partial(e(n-i) - e(n-j))}{\partial w_k}. \quad (3.15)$$

The derivative of the difference in errors appears because of the IP dependence on pairs of samples. Because  $e(n) = z(n) - y(n)$  and  $z(n)$  is independent of the weights this yields

$$\begin{aligned} \nabla_k \hat{V}_2(n) &= \frac{1}{2N^2\sigma^2} \sum_{i=1}^N \sum_{j=1}^N G_{\sqrt{2}\sigma}(e(n-i) - e(n-j)) \\ &\quad \times (e(n-i) - e(n-j))(x_k(n-i) - x_k(n-j)) \end{aligned} \quad (3.16)$$

for  $k = 0, \dots, M-1$ , where  $x_k(n)$  means the sample at tap  $k$  and instant  $n$ .

Notice one important difference of Eq. (3.16) with respect to the steepest descent computation of MSE (Eq. (1.33)): the IP gradient is no longer given by products of inputs and errors, but it is expressed as differences of error samples multiplied by the corresponding differences of inputs weighted by the Gaussian function of the errors. Therefore this algorithm is still local in the topology but it is not local in time. We rewrite Eq. (3.16) below for the adaptation of a FIR with a general kernel  $\kappa$ .

$$\nabla_k \hat{V}_2(n) = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \kappa'_\sigma(e(n-i) - e(n-j))(x_k(n-j) - x_k(n-i)) \quad (3.17)$$

As we can observe, Eqs. (3.16) and (3.17) use a sample block of size  $N$  and so they are equivalent to the steepest descent procedure of MSE, therefore this method is simply called the *MEE algorithm*. The MEE has a complexity of  $O(N^2)$ , and it involves the evaluation of the kernel, which most of the time is a transcendental function. This added complexity is the price paid to extract more structure from the error when compared with the MSE.

The IP gradient expression can be easily extended for arbitrary  $\alpha > 1$  and any kernel following the steps of Chapter 2, and for the FIR case yields

$$\begin{aligned} \nabla_k V_\alpha(n) = & \frac{\alpha - 1}{N^\alpha} \sum_{i=1}^N \left( \sum_{j=1}^N \kappa_\sigma(e(n-i) - e(n-j)) \right)^{\alpha-2} \\ & \times \sum_{j=1}^N \kappa'_\sigma(e(n-i) - e(n-j))(x_k(n-j) - x_k(n-i)). \end{aligned} \quad (3.18)$$

When compared with the theoretical entropy cost of Eq. (3.11), Eq. (3.16) differs in two ways: the first term in Eq. (3.11) resembles Eq. (3.16) in the sense that the expected value is substituted by the empirical mean (the double sums), and that it is multiplied by the input vector when using the chain rule over the topology. However, notice that the estimator works with pairs of samples, so it is not just a matter of substituting  $E \rightarrow \sum$  in Eq. (3.11). Secondly, the second term of Eq. (3.11) does not appear in Eq. (3.16), which is reasonable because when working nonparametrically with the data and kernels, the shape of the PDF is never quantified explicitly. Therefore, we conclude that Eq. (3.16) corresponds to the optimization of EEC in a projected space specified by the kernels for the estimation of the PDF (Chapter 10 shows that effectively the optimization is in the reproducing kernel Hilbert space defined by the kernel).

## Information Forces in Adaptation

In Chapter 2 we provided a physical interpretation of the information potential: when kernels are used for PDF estimation they create a field over the space of the samples defined by the kernel shape. In adaptation the samples that are under study are the error samples  $e(n)$  created by subtracting the system output from the desired response; that is,  $e(n) = z(n) - y(n)$ . Therefore one can also postulate that the samples, interpreted as information particles, create forces among themselves as given by Eq. (2.59). If we compare the information forces with the sensitivity of the cost with respect to the error in Eq. (3.14) we conclude that they are exactly the same. Therefore, in error entropy learning the injected error for adaptation is the information force created by the ensemble on each error sample. Notice that as the IP is a function of pairs of samples, the sensitivity has to be computed with respect to each error, and can be combined as shown in Eq. (3.16).

### 3.4 Minimum Error Entropy Algorithm

We start by providing some important properties of EEC for adaptation and learning. Recall again that filtering is a regression problem in functional spaces and it is the only one treated explicitly in this chapter, although all of the conclusions also apply to regression. The intuition behind the entropy criterion for supervised learning is conceptually straightforward: given samples from an input–output mapping, in order to extract the most structure from the data, the information content of the error signal must be minimized; hence the error entropy over the training dataset must be minimized, which is achieved with a delta function (i.e., with all errors being equal if the system has a sufficient number of degrees of freedom to solve the problem exactly).

**Theorem 3.1.** *The stationary point of the EEC criterion for a linear combiner is achieved when*

$$E \left[ \frac{\partial p_e}{\partial e} X \right] = -E \left[ \frac{\partial p_e}{\partial w} \right]. \quad (3.19)$$

The proof is straightforward from Eq. (3.11). This is an interesting equation, because it imposes a balance between the inner product of the weight vector with the functional derivative of the PDF, and the expected value of the derivative of the PDF with respect to the weights. In the case where the righthand side of Eq. (3.19) is zero, then the derivative of the PDF function becomes orthogonal to the weight vector. We have not investigated under what conditions and classes of PDFs the second term is zero, but this will preserve the orthogonality condition between the gradient of the error PDF and the input subspace as in MSE. In order to fully study Eq. (3.19) a differential geometry approach seems necessary [6]. In fact, because we are dealing with a parameter estimation problem, this result can be interpreted as constraining the inner product of the error’s Fisher information with the weight vector in the data manifold  $(\partial p(e)/\partial e)$  is by definition the Fisher information of the error PDF [6].

**Theorem 3.2.** *The stationary point of the EEC estimated with the IP is translated in an orthogonality condition between the difference in input vectors and the difference in errors weighted by the kernel of the error differences.*

This theorem can also be easily proved from Eq. (3.16). In fact let us first define  $\Delta e_{ij} = e_i - e_j$ ,  $\Delta x_{ij} = x_i - x_j$ . If we express Eq. (3.16) with statistical quantities, then it becomes  $E[G(\Delta e)\Delta e \Delta x]$  and at the stationary point, the differential input and the differential error rotated by  $G(\Delta e)$ , the Gaussian of the differential error, must be orthogonal to each other.

Theorem 3.2 indicates that the MEE provides “similar” weight updates to MSE when the incremental errors are small, but drastically reduces the weight updates when the incremental errors are large. We can therefore expect robustness of the MEE criterion to impulse noise, as detailed in the next section.

**Theorem 3.3.** *In perfect identification, the error entropy criterion is insensitive to additive zero-mean noise independent of the input and desired, regardless of the noise.*

As is well known, the filter weights adapted with MSE are insensitive to zero-mean additive white noise in the desired response [284]. EEC behaves similarly, with some advantages in the small dataset case [87].

*Proof.* Consider the learning process depicted in Figure 3.1. Suppose that the desired signal consists of the superposition of a deterministic component and a zero-mean random component, such that  $z = g(x) + v$ , where  $g(\cdot)$  is the unknown function that the adaptive system is trying to identify and  $v$  is the zero-mean noise with PDF  $p_v(\cdot)$  independent of  $x$ , and  $z$ . Suppose the learning system is a parametric family of functions of the form  $f(x; \mathbf{w})$  where  $\mathbf{w}$  is the vector of parameters, called the weight vector. Assume  $x$ ,  $z$ , and  $y$  are all zero-mean signals without loss of generality. Let  $\mathbf{w}_*$  be the optimal weight vector that minimizes the error entropy, and the error signal be defined as  $e = z - y$ . Let  $\bar{\mathbf{w}}_*$  be the optimal weight vector that minimizes the entropy of the clean error signal that is defined as  $\bar{e} = g(x) - h(x, \mathbf{w})$ . Notice that we have the identity  $e = \bar{e} + v$ . Because  $v$  is an independent noise signal that does not depend on  $w$ , the weights of the adaptive system, when  $\bar{e}$  is  $\delta$ -distributed we have

$$w_* = \arg \min_w H_\alpha(e(w)) = \arg \min_w H_\alpha(\bar{e}(w) + v) = \arg \min_w H_\alpha(\bar{e}(w)) = \bar{w}_*. \quad (3.20)$$

Even if  $\bar{e}$  is not  $\delta$ -distributed (which occurs when the model space does not include the actual system), because the noise and the error are independent  $H_\alpha(\bar{e} + v) \geq H_\alpha(\bar{e})$  and minimizing this upper bound will force the solution to converge to a good value, which would be obtained in the noise-free situation.

*Conjecture 3.1.* *The bandwidth parameter of the Gaussian kernel can be used as a regularizer for the adaptation, mimicking the method of convolution smoothing in global optimization.*

Let us start by remarking that  $\hat{V}(e)$  can be alternatively obtained by convolution of the kernel with the true PDF

$$E[\hat{p}_E(e)] = p_E(e) * \kappa_\sigma(e) = \int p_E(\tau) \kappa_\sigma(e - \tau) d\tau. \quad (3.21)$$

Normally, we select the kernel size with respect to the data properties, but let us look at the problem from the point of view of adaptation. In the space of the system parameters, there is an estimated value  $\hat{V}(e)$  at every point, which is called the performance surface (Figure 3.4). But because  $\hat{V}(e)$  is a function of the kernel size, there is in fact an infinite family of estimates for each point in weight space. Due to Theorem 2.4 and Eq. (3.19) we can expect that the estimate will always be larger than the true quantity. More important,

if the true performance surface is nonconvex, the smoothing properties of the convolution may make it convex as the theory of convolution smoothing indicates [310].

The control of the bandwidth parameter has been proposed as an effective way to smooth local minima in the performance surface created by nonconvex performance functions such as  $V(e)$  [87]. In global optimization the method is called convolution smoothing (CS), and it has been proven effective in many practical applications, such as the adaptation of IIR filters [84]. The basic idea behind this approach is to convolve the cost function with a broad smoothing functional, which initially eliminates the local minima. The width of the smoothing functional can then be gradually decreased until a Dirac- $\delta$  is obtained, which leaves the original cost function. During a proper annealing phase (similar to temperature in simulated annealing [275]), the optimization parameters approach the vicinity of the global optimum and are in the domain of attraction that yield the optimal solution by gradient descent.

The interesting aspect of convolution smoothing for ITL is that normally the smoothing function has to be applied to the cost function after the fact, but in ITL the cost function  $\hat{V}(e)$  is created through a convolution, so the method is intrinsic to the ITL formalism; we just need to use the kernel size to achieve the optimization goal. However, in ITL the kernel size cannot be decreased to zero, otherwise the estimation of  $V(e)$  by the IP breaks down. Therefore the minimum value of the kernel size should be dictated by Silverman's or equivalent rule. This value will not affect the location of the minimum in weight space in perfect identification. Unfortunately, we have not been able to prove all of the conditions of the theory of convolution smoothing applied to ITL, but experimental results have shown that annealing the kernel size during adaptation helps the convergence to the global minimum [87, 165, 223]. Appendix A summarizes current knowledge about kernel size annealing.

## 3.5 Analysis of MEE Performance Surface

### General Shape of EEC Performance Surface

It is well known that the MSE cost function is a paraboloid facing up in the space of the free parameters of the FIR filter. Here we analyze the overall shape of the MEE cost function for Gaussian kernels, and we start with a single-parameter FIR; that is,  $y(n) = wx(n)$ . Because  $e(n - i) = z(n - i) - y(n - i)$  Eq. (3.12) can be written

$$\begin{aligned} \hat{V}_2(E, w) &= \frac{1}{N^2 \sqrt{2\pi}\sigma} \sum_{i=1}^N \sum_{j=1}^N e^{-[(\Delta z_{ij} - w\Delta x_{ij})/2\sigma]^2} \\ &= \frac{1}{N^2 \sqrt{2\pi}\sigma} \sum_{i=1}^N \sum_{j=1}^N e^{-(\Delta z_{ij}/2\sigma)^2} e^{+(2w\Delta x_{ij}\Delta z_{ij}/2\sigma)} e^{-(w\Delta x_{ij}/2\sigma)^2}, \end{aligned} \quad (3.22)$$

where we use the notation  $\Delta z_{ij} = z_i - z_j$ ,  $\Delta x_{ij} = x_i - x_j$ . This expression is not amenable to a simple interpretation but there are several important observations.

- All the terms in the exponent appear divided by the kernel size, which means that the scale in the performance function space is dictated by the kernel size. Or, in other words, the concept of “large” or “small” errors in adaptation is always going to be relative to the kernel size.
- The first exponential term in the double sum is negative, independent of the weight, and so it will be a constant dictated simply by the desired signal; the second term is positive, linear in the weights, and contains the cross-product difference between input and desired response; and the third term is negative, depends only on the input difference, and is a function of  $w^2$ .
- Due to the square dependence on the weight, the third term will dominate for large positive or negative values and we can expect the performance surface to peak at some point in weight space (dictated primarily by the second term) and then decrease exponentially with increasing or decreasing  $w$  with a slight asymmetry due to the second term. The rate of decrease is solely controlled by the input signal.

At first, adaptation of the EEC cost function seems to be unrelated to what we know for MSE, but there is a hidden similarity. In fact, if we do an approximation of the exponential function truncated at the second term, which is only valid for small errors, we see that  $e^{-x^2} \sim 1 - x^2$ ; that is, we obtain a quadratic function of the error, but now the parabola is facing down, meaning the optimization is for a maximum. What this means is that near the optimum there is always a neighborhood (controlled by the kernel size) where the EEC performance surface is well approximated by a quadratic function. However, this does not imply that the stationary point of the EEC coincides with the MSE in weight space. To see this let us compute the optimal weight for the one-parameter case. Taking the derivative of Eq. (3.22) and equating it to zero we obtain

$$w^* = \frac{\sum_{i=1}^N \sum_{j=1}^N G_\sigma(\Delta e_{ij}) \Delta z_{ij} \Delta x_{ij}}{\sum_{i=1}^N \sum_{j=1}^N G_\sigma(\Delta e_{ij}) \Delta x_{ij}^2}. \quad (3.23)$$

If we recall, the least square solution for the single-parameter case is  $w = \sum_{i=1}^N z_i x_i / \sum_{i=1}^N x_i^2$ . If one substitutes the variables with their time increments (referred to as the delta) the form of the solution is very similar, except that each term in the sum is multiplied by the Gaussian of the error difference. In perfect identification the errors go to zero, so the Gaussian goes to one, and we can expect that the two solutions coincide, provided the input or desired response are zero-mean stationary random processes. The latter condition is required to preserve the optimal weight vector between the MSE solution and the formulation with the differences of input/desired response pairs

because  $\sum_i \sum_j \Delta z_{ij} \Delta x_{ij} = \sum_i x_i z_i + \sum_j x_j z_j - \sum_i x_i \sum_j z_j - \sum_i z_i \sum_j x_j$  and the two last sums are zero for zero mean signals (likewise for the denominator).

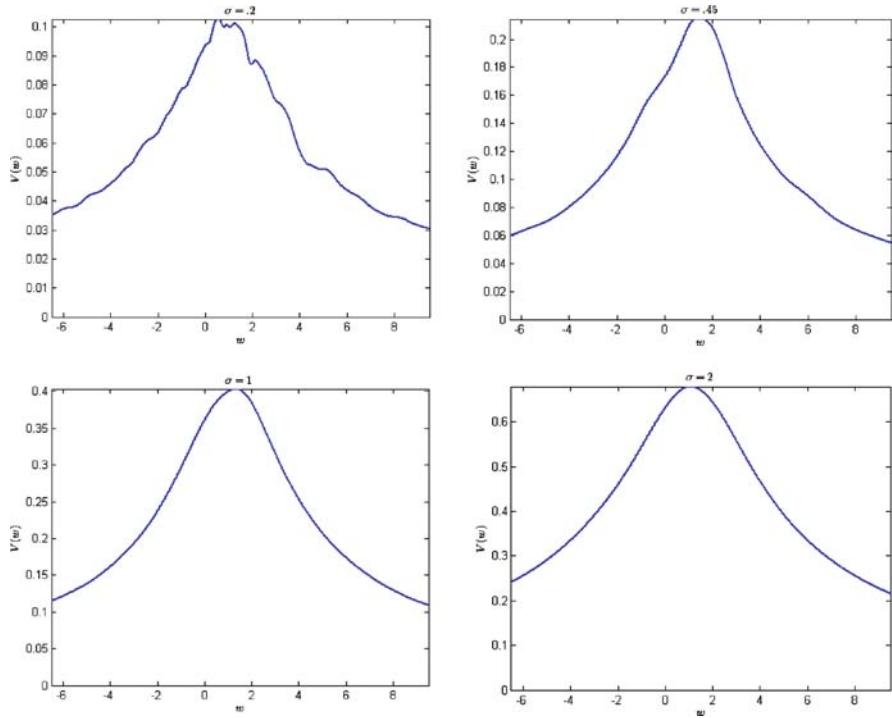
The other case where the two solutions are similar is when the kernel size is chosen much larger than the one required from the data structure (and the input or desired response data are zero-mean), which means that the Gaussian of the errors become approximately independent of the index and can be factored out of the sums. Alternatively, this can be interpreted as stating that the quadratic approximation of the EEC cost includes the full parameter space.

In general, the EEC minimum will be different from the MSE in parameter space, and the EEC shape does not display constant curvature (unlike the MSE cost function) because  $G_\sigma(\Delta e_{ij})$  changes across iterations and affects all the terms in the sum differently, starting with small values (large error differences) and increasing until all the errors are basically the same (the minimum of entropy). These ideas are made more precise with a simulation below.

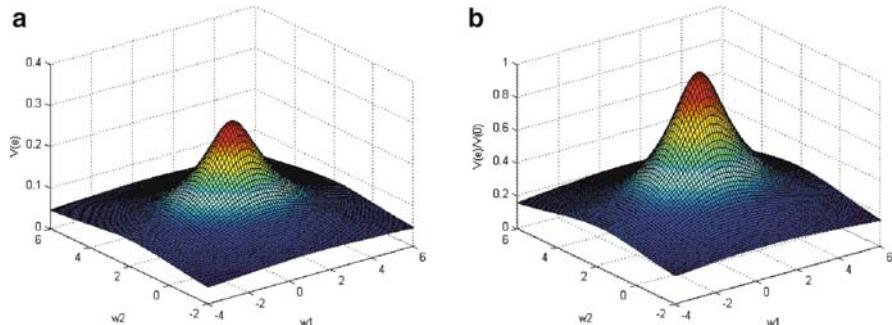
The simulation adapts a 1-tap FIR in a system identification configuration (Figure 3.1), for which the desired response data are generated by a 2-tap FIR with weight vector  $w_* = [1, 2]^T$ . There is no measurement noise and the input to the filter is white Gaussian noise with zero-mean and unit variance. This experimental setup is very simple but we can visualize the weight tracks during adaptation to evaluate convergence and we know that the stationary point is equal to the MSE cost because the system identification solution is unique. On the other hand it does not fully display all the characteristics of the EEC, which in general will provide different optimal solutions that may be “better” than the MSE errors under some conditions.

Our first simulation is obtained with a single-tap FIR filter, that is,  $y(n) = w_0 x(n)$ , and Figure 3.4 depicts the performance surface for various values of the kernel size (according to Silverman’s rule, the kernel size should be  $\sigma = 0.45$  for this specific example). Notice that the cost function displays local minima for small values of  $\sigma$ , but it becomes progressively smoother, symmetric and closer to a quadratic when  $\sigma$  grows.

Figure 3.5 shows an example of the information potential and the normalized IP, that is,  $(V(e)/V(0))$  for a two-tap filter that is able to identify the unknown system exactly (i.e., zero error). The maximum value of the normalized IP for perfect identification is 1, but in all other cases  $(V(0) - V(E))/V(0)$  can be interpreted as the deviation from the best possible solution (i.e., similar to the final error power in MSE). Recall that maximizing IP corresponds to minimizing Renyi’s entropy. Notice that the IP is always positive and pretty flat in most of the space, which is very different from the MSE cost function. In regions close to the peak, the surface is steeper than a quadratic function.



**Fig. 3.4.** The performance surface for four values of the kernel size.



**Fig. 3.5.** Comparison of the IP and normalized IP ( $\sigma = 1$ ): (a) information potential  $[V(e)]$ ; (b) normalized information potential  $[V(e)/V(0)]$

### Multiple Parameter Analysis of EEC Optimal Solution

An analysis of Eq. (3.23) shows that it is possible to extend the closed form optimal weight solution for filters with multiple parameters (weights), when the kernel is Gaussian, just by extending the least square approach to EEC.

In fact, assume now that the filter has  $M$  parameters. Taking the derivative of Eq. (3.12) with respect to the weights yields Eq. (3.16). Substituting  $\Delta e_{i,j} = \Delta z_{i,j} - \sum_{l=0}^{M-1} w_l \Delta x_{i-l,j-l}$  in this equation and equating to zero obtains

$$\begin{aligned} \sum_{i=1}^N \sum_{j=1}^N G_\sigma(\Delta e_{ij}) \Delta z_{ij} \Delta x_{i-k,j-k} &= \sum_{l=0}^{M-1} w_l \sum_{i=1}^N \\ \sum_{j=1}^N G_\sigma(\Delta e_{ij}) \Delta x_{i-l,j-l} \Delta x_{i-k,j-k} &\quad k = 0, \dots, M-1 \end{aligned} \quad (3.24)$$

Note the similarity of this set of  $M$  equations with  $M$  unknown filter parameters to the least square solution presented in Chapter 1: we have the cross-correlation between the incremental desired and the incremental input (which corresponds to the cross-correlation vector in least squares) equal to the autocorrelation matrix of the incremental input, each weighted by the Gaussian of the incremental errors; that is, in vector form,  $\mathbf{R}_\Delta \mathbf{w}^* = \mathbf{p}_\Delta$ , where  $\mathbf{R}$  is the  $M \times M$  locally Gaussian averaged delta autocorrelation function with elements  $R_\Delta(w_{l,k}) = \sum_{i=1}^N \sum_{j=1}^N G_\sigma(\Delta e_{ij}) \Delta x_{i-l,j-l} \Delta x_{i-k,j-k}$  and the  $M$ -dimensional cross-correlation vector of locally Gaussian averaged incremental inputs and desired signals with elements  $p_\Delta(w_k) = \sum_{i=1}^N \sum_{j=1}^N G_\sigma(\Delta e_{ij}) \Delta z_{ij} \Delta x_{i-k,j-k}$ .

Note also the differences. In least squares, the solution is only a function of the external system variables, whereas here these equations are still a function of the error signal, which in itself is a function of the filter parameters. Therefore, Eq. (3.24) should not be interpreted as an analytic solution, but more as a fixed point update algorithm. It is readily computable provided we have access to the error and it has no free parameters. This means that once the new parameters are estimated, we estimate the new error to plug in Eq. (3.24) to get the new parameters in recursive fashion. Note the effect of the Gaussian kernel that is multiplying all the elements of both the delta autocorrelation and delta cross-correlation functions, and is what embodies the difference between straight least squares and this weighted version of least squares. Again we can expect the EEC solution to match least squares for zero mean input or zero mean desired signals in perfect identification, that is, when the error is zero. We continue to analyze this solution closely below and also in Chapter 4.

### Analysis of the Gradient Around the EEC Optimal Solution

As is well known in adaptation using the MSE criterion, the stepsize (or learning rate) for convergence is linked to the largest eigenvalue of the input autocorrelation matrix. In practice the inverse of the input power is considered as a stricter, but much easier to estimate, upper bound. If we analyze Eq. (3.16) closely we can see that the expression is a much more complex function of the error, unlike the steepest descent algorithm. Therefore, our first problem is to estimate the shape and steepness of the information potential cost function to arrive at an understanding of how to select the stepsize.

Suppose that the adaptive system under consideration in Figure 3.1 is a linear combiner with a weight vector  $\mathbf{w}$ . The error samples are  $e(n) = z(n) - \mathbf{w}^T \mathbf{x}(n)$ , where  $\mathbf{x}(n)$  is the input vector at sample  $n$ , formed by feeding the input signal to a tapped delay line (FIR filter). The gradient of the information potential estimator with respect to the weight vector in vector notation is simply

$$\frac{\partial V_\alpha}{\partial \mathbf{w}} = \frac{(\alpha - 1)}{N^\alpha} \sum_j \left( \sum_i \kappa_\sigma(e_j - e_i) \right)^{\alpha-2} \cdot \left( \sum_i \kappa'_\sigma(e_j - e_i)(\mathbf{x}_i - \mathbf{x}_j)^T \right). \quad (3.25)$$

In this expression, further simplifications are possible through the use of the scaling property of the kernel size and the following identity between the derivatives of a width- $\sigma$  kernel and a unit-width kernel  $\kappa'_\sigma(x) = 1/\sigma^2 \kappa'(x/\sigma)$ . With these substitutions, the explicit expression for the gradient is easily determined to be

$$\frac{\partial V_\alpha}{\partial \mathbf{w}} = \frac{(\alpha - 1)}{\sigma^\alpha N^\alpha} \sum_j \left( \sum_i \kappa(\Delta e_{ji,w}) \right)^{\alpha-2} \cdot \left( \sum_i \kappa'(\Delta e_{ji,w}) \cdot (\mathbf{x}_i - \mathbf{x}_j)^T \right), \quad (3.26)$$

where  $\Delta e_{ji,w} = (z_j - z_i) - \mathbf{w}^T (\mathbf{x}_j - \mathbf{x}_i)$ .

To continue with our analysis of adaptation near the optimum, we consider the Taylor series expansion truncated to the linear term of the gradient around the optimal weight vector  $\mathbf{w}_*$ .

$$\nabla \hat{V}_\alpha(\mathbf{w}) \sim \nabla \hat{V}_\alpha(\mathbf{w}_*) + \frac{\partial \nabla \hat{V}_\alpha(\mathbf{w}_*)}{\partial \mathbf{w}} (\mathbf{w} - \mathbf{w}_*). \quad (3.27)$$

Notice that truncating the gradient at the linear term corresponds to approximating the cost function around the optimal point by a quadratic function. From the previous discussion there is always a neighborhood of the optimum controlled by the kernel size where this approximation is valid. The Hessian matrix of this quadratic performance surface approximation is  $\Xi/2$ , where  $\Xi$  is given as

$$\begin{aligned} \Xi &= \frac{\partial^2 \hat{V}_\alpha(\mathbf{w}_*)}{\partial \mathbf{w}^2} = \frac{\partial^2 \hat{V}_\alpha(\mathbf{w}_*)}{\partial \mathbf{w}^2} = \frac{(\alpha - 1)}{\sigma^\alpha N^\alpha} \sum_j \left[ \sum_i \kappa(\Delta e_{ji,w_*}) \right]^{\alpha-3} \cdot \\ &\left\{ (\alpha - 2) \left[ \sum_i \kappa'(\Delta e_{ji,w_*}) \cdot (\mathbf{x}_i - \mathbf{x}_j) \right] \cdot \left[ \sum_i \kappa'(\Delta e_{ji,w_*}) \cdot (\mathbf{x}_i - \mathbf{x}_j)^T \right] \right\} \\ &+ \left[ \sum_i \kappa(\Delta e_{ji,w_*}) \right] \cdot \left[ \sum_i \kappa''(\Delta e_{ji,w_*}) \cdot (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T \right] \end{aligned} \quad (3.28)$$

We can see that even under quadratic approximation, the EEC Hessian matrix is a function of the input and the desired response through the error,

unlike the MSE cost function. Moreover, due to the inclusion of the errors, it is also not constant curvature. However, one can state that if the adaptation problem can be solved with small final error, then the effect of the desired response on the shape of the performance surface decreases proportionally. In parallel to the work in MSE, we can define a new weight vector space  $\bar{\mathbf{w}} = \mathbf{w} - \mathbf{w}_*$  whose origin is translated to the optimal solution  $\mathbf{w}_*$ , and rewrite the linearized dynamics of the weight equations in the vicinity of the solution in terms of the stepsize and the Hessian matrix as  $\dot{\bar{\mathbf{w}}}(n+1) = [\mathbf{I} + \eta \Xi] \bar{\mathbf{w}}(n)$ . These are the coupled equations for the translated weights. In order to obtain decoupled equations, we rotate the vector space by defining  $\mathbf{v} = \mathbf{Q}^T \bar{\mathbf{w}}$ ,  $\mathbf{Q}$  being the orthonormal (modal) matrix consisting of the eigenvectors of  $\Xi$ . Thus, the uncoupled dynamics for the translated and rotated weights are  $\dot{\mathbf{v}}(n+1) = [\mathbf{I} + \eta \Lambda] \mathbf{v}(n)$  where  $\Lambda$  is the diagonal eigenvalue matrix with entries ordered in correspondence with the ordering in  $\mathbf{Q}$ . From this set of equations, we can isolate the dynamics of the weight vector along each mode of the matrix  $\Xi$ . Specifically, for the  $i$ th mode, the dynamic equation will only depend on the  $i$ th eigenvalue of  $\Xi$  by

$$v_i(n+1) = [1 + \eta \lambda_i] v_i(n), \quad i = 1, \dots, l \quad (3.29)$$

Note that, because  $\Xi$  is the Hessian of the performance surface evaluated at a local maximum its eigenvalues are negative. For stable dynamics, all of the coefficients in the  $n$  equations of Eq. (3.29) must be inside the unit circle; that is,  $|1 + \eta \lambda_i| < 1$ . For stability this yields the following bound for the stepsize

$$0 < \eta < \frac{1}{\max_i |\lambda_i|}. \quad (3.30)$$

As expected, this condition is the same as the MSE criterion [143], except we consider the eigenvalues of the Hessian matrix of the second-order approximation of the information potential instead of those of the covariance matrix (autocorrelation matrix in the FIR filter case) of the input vector to the FIR. Unfortunately, as seen in Eq. (3.28) it is very difficult to progress towards a reasonable approximation of the eigenvalue matrix to decide about a reasonable stepsize. Using the power of the input as an upper bound for the largest eigenvalue still works under most conditions.

At this point, it also becomes possible to talk about time constants of the modes in the neighborhood of the optimum point. We can determine an approximate time constant for each individual mode whose dynamic equations are governed by Eq. (3.29). Specifically, for the  $k$ th mode, we write  $(1 + \eta \lambda_k) = e^{-1/\tau_k}$  from which the time constant is determined to be

$$\tau_k = \frac{-1}{\ln(1 + \eta \lambda_k)} \approx \frac{-1}{\eta \lambda_k} = \frac{1}{\eta |\lambda_k|}. \quad (3.31)$$

The time constants allow us to compare the convergence times of different modes. In order to evaluate the overall convergence speed, one must consider

the slowest mode, which corresponds to the largest time constant, that is, the smallest eigenvalue. Understanding the relationship between the eigenvalues of the Hessian matrix in Eq. (3.28) and the two parameters, the kernel size and the entropy order, is crucial to maintaining the stability of the algorithm following any changes in these parameters. One practical case where this relationship becomes important is when we adapt the kernel size during the training in connection with Conjecture 3.1. Because in this approach the kernel size is decreased, we need to know how to adapt the stepsize to achieve faster learning in the initial phase of adaptation (by using a larger stepsize) and stable convergence in the final phase (by using a smaller stepsize). As an example, consider the case where we evaluate the quadratic information potential using Gaussian kernels. In this case, the Hessian matrix simplifies to

$$\Xi = \frac{1}{\sigma^2 N^2} \sum_j \left[ \sum_i \kappa''(\Delta e_{ji, w_*}) (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T \right]. \quad (3.32)$$

Observe from Eq. (3.32) that as  $\sigma$  increases,  $\Delta e_{ji, w_*} \rightarrow 0$ , therefore,  $\kappa''(\Delta e_{ji, w_*}) \rightarrow 0^-$  with speed  $O(\sigma^{-6})$ . This is faster than the reduction rate of the denominator, which is  $O(\sigma^{-2})$ , hence overall, the eigenvalues of  $\Xi$  approach  $0^-$ . This means that the crest near the global maximum gets wider and one can use a larger stepsize in steepest ascent, while still achieving stable convergence to the optimal solution. In fact, this result can be generalized to any kernel function and any  $\alpha$ . This analysis is useful, but the big challenge that remains is to estimate easily from the data an upper bound for the eigenvalues of the quadratic approximation to select the stepsize.

### Analysis of the Eigenvalues for Varying $\alpha$

A precise analysis cannot be analytically pursued, but we can still predict how the eigenvalues of the Hessian behave as this parameter is modified. In order to estimate the behavior of the eigenvalues under changing  $\alpha$ , we exploit the following well-known result from linear algebra relating the eigenvalues of a matrix to its trace. For any matrix  $\mathbf{R}$ , whose eigenvalues are given by the set  $\{\lambda_i\}$ ,  $\sum_i \lambda_i = \text{trace}(\mathbf{R})$ . Now consider the general expression of  $\Xi$  given in Eq. (3.32). The trace of  $\Xi$  is computed as

$$\begin{aligned} \text{trace}(\Xi) &= \frac{(\alpha - 1)}{\sigma^\alpha N^\alpha} \sum_j \left[ \sum_i \kappa(\Delta e_{ji, w_*}) \right]^{\alpha-3} \\ &\times \left\{ (\alpha - 2) \sum_k \left[ \sum_i \kappa'(\Delta e_{ji, w_*}) \cdot (x_{ik} - x_{jk}) \right]^2 \right. \\ &\left. + \left[ \sum_i \kappa(\Delta e_{ji, w_*}) \right] \cdot \left[ \sum_i \kappa''(\Delta e_{ji, w_*}) \cdot \left( \sum_k (x_{ik} - x_{jk})^2 \right) \right] \right\} \end{aligned} \quad (3.33)$$

The eigenvalues of  $\Xi$  are negative and the dominant component, which introduces this negativity, is the term in the last line of Eq. (3.33). The negativity arises naturally because we use a differentiable symmetric kernel; at  $\mathbf{w}_*$  the entropy is small, therefore the error samples are close to each other and the second derivative evaluates as a negative coefficient. Now let's focus on the term that involves the  $(\alpha - 3)$ -power in the first line of Eq. (3.33). All other terms vary linearly with  $\alpha$ , thus this term dominantly affects the behavior of the trace when  $\alpha$  is varied. Consider the case where  $\sigma$  is small enough such that the small entropy causes the kernel evaluations in the brackets to be close to their maximum possible values and the sum therefore exceeds one. In this case, the power of the quantity in the brackets increases exponentially with increasing  $\alpha$  (for  $\alpha > 3$ ), thus regardless of the terms affected linearly by  $\alpha$ , the overall trace value decreases (increases) in absolute value. Consequently, a narrower crest towards the maximum appears and the upper bound on the stepsize for stability is reduced.

On the other hand, if the kernel size is large so that the sum in the brackets is less than one, then the  $(\alpha - 3)$ -power of this quantity decreases, thus resulting in a wider crest towards the maximum in contrast to the previous case (for  $\alpha > 3$ ). However, in practice we do not want to use a very small or a very large kernel size, as this will increase the variance or increase the bias of the Parzen estimation, respectively.

Another important observation is to evaluate the changes of the trace across iterations for constant  $\alpha$  and kernel size. Eq. (3.33) is rather complex, but notice that all the input factors are multiplied by kernels and their first and second derivative. When the error is large, all these kernel evaluations will be close to zero so we can expect that the maximum eigenvalue of  $\Xi$  will be much smaller than the one obtained from the autocorrelation matrix far from the optimum. However, notice that near the optimum, the terms multiplying  $\kappa'$  are going to be small, whereas the ones involving  $\kappa$  are close to 1 and those for  $\kappa''$  have a large negative value that is inversely proportional to the kernel size. Therefore, the trace around the optimum may be larger than for the autocorrelation function. We can expect that the maximum eigenvalue of the EEC cost is far from constant during adaptation, which indirectly translates the fact that the performance surface is not a paraboloid as Figure 3.4 shows. Therefore, adaptive stepsize algorithms seem particularly appropriate for the IP cost function. These conclusions are summarized in the following three facts.

**Fact 3.1.** Regardless of the entropy order, increasing the kernel size results in a wider crest around the optimal solution because the absolute values of the (negative) eigenvalues of the IP Hessian matrix decrease.

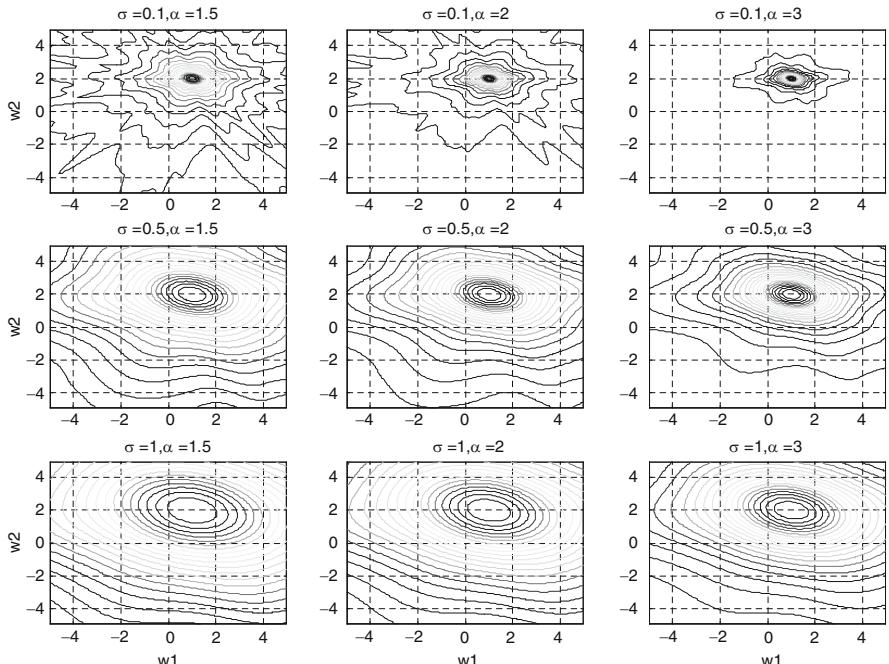
**Fact 3.2.** The effect of entropy order on the eigenvalues of the Hessian depends on the value of the kernel size. If the kernel size is small, then increasing the entropy order increases the absolute values of the (negative) eigenvalues of the Hessian of the information potential function at the global maximum.

This results in a narrower crest. If the kernel size is large, the effect is the opposite; increasing the entropy order decreases the absolute value of the eigenvalues of the Hessian of the information potential, resulting in a wider crest. This analysis is expected to hold at least for  $\alpha > 3$ .

**Fact 3.3.** The largest eigenvalue across iterations for constant  $\alpha$  and kernel size is far from constant, reflecting the fact that the IP performance surface is not a quadratic function. To avoid divergence, the stepsize should be set smaller than the input power, but we can expect slow convergence far from the optimum.

We remark that our conclusions in this section do not only apply to the eigenvalues of  $\boldsymbol{\Xi}$ , but they generalize to how these two parameters affect the volume of the region where our quadratic approximation is valid. These results are very useful from a practical point of view, because they explain how the structure of the performance surface can be manipulated by adjusting these parameters. Besides, they identify the procedures to adjust the stepsize for fast and stable convergence.

Figure 3.6 depicts the effect of entropy order and kernel size on the performance surface when the estimator in Eq. (3.26) is utilized in the simple



**Fig. 3.6.** Contours of information potential in supervised FIR training for various choices of kernel size ( $\sigma$ ) and entropy order ( $\alpha$ ) (from [88]).

problem of Section 3.5. Evaluation of associated gradients and Hessians is carried out using the formulas presented in the preceding sections. This case study aims to illustrate how the performance surface (here represented by its contour plots) of the information potential criterion for supervised training of a linear combiner is altered as a consequence of changing entropy order and kernel size in the estimator.

Recall that we have concluded that as the kernel size is increased, the valley around the global maximum becomes wider (allowing a larger step-size for stable convergence) as well as the volume of the region of quadratic approximation. This is clearly observed in the columns of Figure 3.6. As we predicted the coverage area of the quadratic approximation expands as the cost function approaches MSE when the kernel size is increased. In Figure 3.6, each row represents a constant kernel size ( $\sigma = 0.1, 0.5, 1$ ) and each column represents a constant entropy order ( $\alpha = 1.5, 2, 3$ ), respectively.

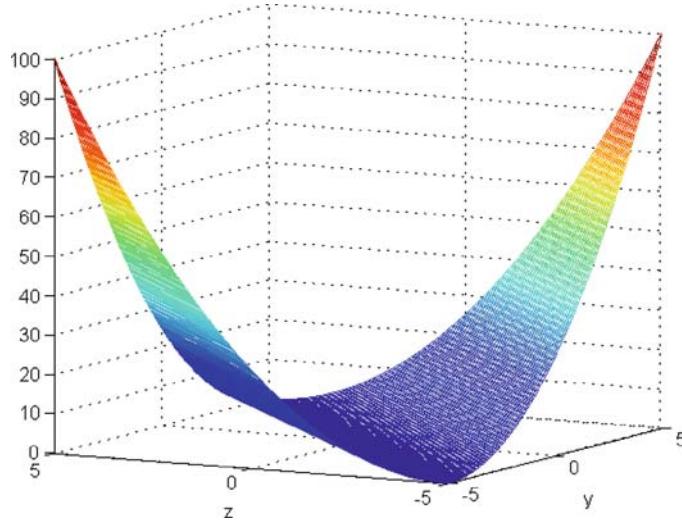
### 3.6 Error Entropy, Correntropy, and M Estimation

The usual goal in adaptive filtering and regression (Figure 3.1) is to bring the system output as “close” to the desired signal as possible. The concept of “close” implicitly or explicitly employs a distance function or similarity measure. MSE is probably the most widely used cost function and it has the obvious meaning of an  $L_2$  distance if we replace the statistical expectation with the sample mean estimator. Assume that the desired signal and the system output are random variables  $Z = \{z_i\}$ ,  $i = 1, \dots, N$ , and  $Y = \{y_i\}$ ,  $i = 1, \dots, N$ , respectively, from which we obtain  $N$  samples from our experiment, and we define a new random variable  $E = Z - Y$ . The mean square error (MSE) is defined as

$$MSE(Y, Z) = E[(Y - Z)^2] = \int_y \int_z (y - z)^2 p_{YZ}(y, z) dy dz = \int_e e^2 p_E(e) de \quad (3.34)$$

where the error square term is illustrated in Figure 3.7.

Notice that MSE is a quadratic function in the joint space with a valley along the  $[z = y]$  line. Because similarity quantifies how close  $Z$  is from  $Y$  in probability, this intuitively explains why MSE is a similarity measure in the joint space. In Eq. (3.34), we also see that the error square is weighted by the PDF of the error. However, the quadratic increase of the error for values away from the  $z = y$  line due to the second moment has the net effect of amplifying the contribution of samples that are far away from the mean value of the error distribution and it is why Gaussian distributed residuals (or other short-tail distributions) provide optimality for the MSE procedure. But it is also the reason why other fat-tail data distributions such as the Laplacian and in particular error distributions with outliers, nonsymmetric,



**Fig. 3.7.** The MSE cost function in the joint space.

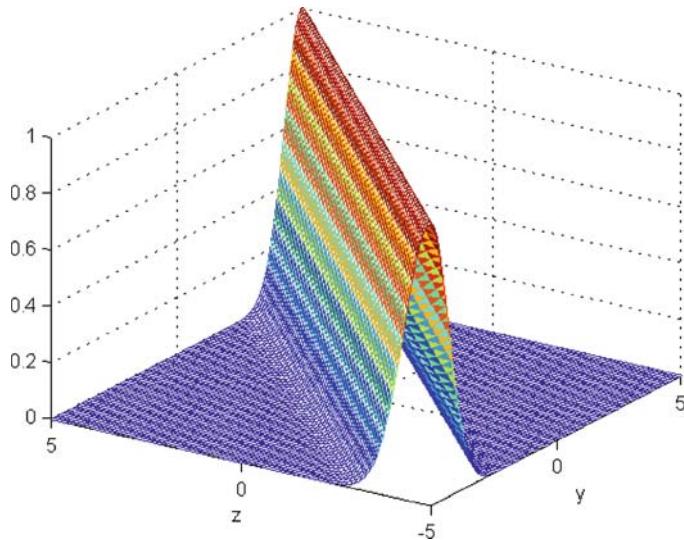
or nonzero-mean, make the MSE suboptimal. This has been well recognized in the statistical literature [157], and methodologies that go by the name robust statistics are preferred in this scenario. We show below that the 2-norm of the PDF and the IP estimator play a very central role in implementing robust statistics.

We use the definitions of quadratic Renyi's entropy and the estimator of the IP of Eq. (3.12) exclusively using the Gaussian kernel, although the results extend to other kernels. The EEC is a well-defined cost function in the sense that it has a global solution manifold where all error entries are equal (to obtain a single point the mean error value must be constrained, for instance, by using the mean of the desired response; see Section 3.3). However, does EEC define a metric in the same sense that MSE defines an  $L_2$  norm on the error sample space? This is the question that we will address in this section. In order to answer this question we have to introduce a related cost function for learning named the *error correntropy criterion* (ECC) [201] defined by a function of two arguments called *cross-correntropy*. Chapter 10 studies correntropy extensively, and here we just use the concept to motivate robust estimation. Cross-correntropy for two random variables  $Z$  and  $Y$  is formally defined as

$$v(Z, Y) = E_{ZY}[G_\sigma(Z - Y)] = \int \int G_\sigma(z - y)p(z, y)dzdy. \quad (3.35)$$

When sampling from the densities, cross-correntropy can be estimated as

$$\hat{v}(Z, Y) = \frac{1}{N} \sum_{i=1}^N G_\sigma(z_i - y_i) = \frac{1}{N} \sum_{i=1}^N G_\sigma(e_i). \quad (3.36)$$



**Fig. 3.8.** Correntropy in the joint space.

Figure 3.8 shows a plot of cross-correntropy for the Gaussian kernel with  $\sigma = 1$ . As one observes, cross-correntropy can also be used as a similarity measure in the joint space, but it differs appreciably from the MSE shown in Figure 3.7. In fact, the cost emphasizes the behavior between  $Z$  and  $Y$  along the line  $z = y$ , and exponentially attenuates contributions away from this line depending on the shape and parameter of the kernel utilized.

From Eq. (3.36), one can propose the ECC as a new cost function for adaptation well grounded in statistical meaning that maximizes the error probability density at the origin and that yield the maximum correntropy criterion (MCC) algorithm; that is,

$$MCC = \max_w \hat{v}(E),$$

where the parameters  $w$  control the error PDF  $E = Z - Y$ . Indeed, using the Parzen method, the error PDF  $p_E(e)$  can be estimated as

$$\hat{p}_E(e) = \frac{1}{N} \sum_{i=1}^N G_\sigma(e - e_i).$$

Evaluating this PDF at  $e = 0$ , we obtain, comparing with Eq. (3.36),

$$\hat{v}(Z, Y) = \hat{p}_E(0). \quad (3.37)$$

Therefore, MCC effectively increases the value of the error PDF at zero (if possible a delta function), which is the natural thing to do in regression or adaptive filtering, where the goal is to increase the number of small deviations between  $Z$  and  $Y$ .



If we recall the EEC discussion early in this chapter, we also see that maximizing the information potential was achieved by creating a PDF that was very peaky (if possible a delta function), but we lost the ability to center it at zero because we were simply maximizing the mean of the transformed error variable. These interpretations are very important to understand these three cost functions (i.e., ECC, EEC, and MSE) and their effects on learning.

## A Brief Review of M Estimation

The MSE cost function performs very well when the error statistics are zero-mean and the PDF of the noise is Gaussian or short-tailed. In many practical conditions these conditions are violated by outliers that can make the noise nonzero-mean and extend the tails of the distribution. This is well recognized in statistics, and there are many possible ways of mitigating outliers, either by removing them manually, using trimming methods or using other statistics that are more resilient to noise such as the rank statistics (for which the median is the best-known example) [297]. An alternative that has many followers is the idea of weighted least squares, where the second moment of the error is substituted by other less steeply increasing functions of the error that are customized for the application (linear increase, windowizing, even saturating after some value as the bisquare proposed by Tukey) [127]. The solution is less sensitive to outliers, therefore it is called robust.

A systematic way of handling outliers is achieved by introducing the concept of robustness in maximum likelihood as proposed by Huber [157]. Recall that the maximum likelihood estimator from a data observation  $x_1, \dots, x_n$  assuming the distribution  $p(x)$  is known except for the parameters  $\theta$ , can be written as

$$\hat{\theta}_{MLE} = \arg \max_{\theta} p(x_1, \dots, x_N | \theta). \quad (3.38)$$

If we assume i.i.d. observations we can rewrite as

$$\begin{aligned} \hat{\theta}_{MLE} &= \arg \max_{\theta} \prod_{i=1}^N p(x_i | \theta) = \arg \max_{\theta} \sum_{i=1}^N \log p(x_i | \theta) \\ &= \arg \min_{\theta} \sum_{i=1}^N (-\log p(x_i | \theta)), \end{aligned} \quad (3.39)$$

due to the monotonicity of the logarithm. Huber defined the M-estimators as a generalization of maximum likelihood as

$$\min_{\theta} \sum_{i=1}^N \rho(x_i) \quad \text{or} \quad \sum_{i=1}^N \psi(x_i) = 0 \quad \text{with} \quad \psi(x) = \frac{d\rho(x)}{dx}, \quad (3.40)$$

where  $\rho(x)$  must obey the properties:

1.  $\rho(x) \geq 0$ .
  2.  $\rho(0) = 0$ .
  3.  $\rho(x) = \rho(-x)$ .
  4.  $\rho(x_i) \geq \rho(x_j)$ ,  $|x_i| > |x_j|$ .
- (3.41)

This estimator can be readily applied to regression problems in lieu of the MSE criterion. Let us assume the data model  $z = \mathbf{w}^T \mathbf{x} + e$ . The robust cost function can be applied to the error in Figure 3.7 yielding

$$J(e) = \sum_{i=1}^N \rho(e_i) = \sum_{i=1}^N \rho(z_i - \mathbf{w}^T \mathbf{x}_i). \quad (3.42)$$

Taking the derivative of the cost w.r.t. the parameter vector  $\mathbf{w}$  and equating to zero produces a set of equations of the form,

$$\sum_{i=1}^N \psi(z_i - \mathbf{w}^T \mathbf{x}_i) \mathbf{x}_i^T = 0.$$

Define the weighting function  $\gamma(e) = \psi(e)/e$  and let  $\gamma_i = \gamma(e_i)$ . Then the equivalent gradient becomes

$$\sum_{i=1}^N \gamma_i \mathbf{x}_i^T e_i = 0, \quad (3.43)$$

which corresponds to a weighted least squares cost function inasmuch as  $\min_w \sum_{i=1}^N \gamma_i e_i^2$ , but it requires an iterative reweighted least squares solution since both the weighting function and the parameters depend upon the residues  $e$  [104]. This result establishes formally the link between M-estimation and weighted least squares, which includes least squares as a special case; that is,  $\rho_{LS}(e) = e^2$  and  $\gamma(e) = 1$  (see Table 3.1).

**Table 3.1.** Comparison of Robust Least Square Estimators

Method	Cost Function	Weighting Function
Least squares	$\rho_{LS}(e) = e^2$	$\gamma_{LS}(e) = 1$
Huber	$\rho_H(e) = \begin{cases} 1/2e^2 &  e  < \alpha \\ \alpha e  - 1/2k^2 &  e  > \alpha \end{cases}$	$\gamma_H(e) = \begin{cases} 1 &  e  < \alpha \\ \alpha/ e  &  e  > \alpha \end{cases}$
Bi-square	$\rho_{Bi}(e) = \begin{cases} \alpha^2/6(1 - (1 - (e/\alpha)^2)^3) &  e  < \alpha \\ \alpha^2/6 &  e  > \alpha \end{cases}$	$\gamma_{Bi}(e) = \begin{cases} (1 - (e/\alpha)^2)^2 &  e  < \alpha \\ 0 &  e  > \alpha \end{cases}$
ECC	$\rho_{CEC}(e) = (1 - \exp(-e^2/2\sigma^2))/\sqrt{2\pi}\sigma$	$\gamma_{CEC}(e) = \exp(-e^2/2\sigma^2)/\sqrt{2\pi}\sigma^3$

### 3.7 Correntropy Induced Metric and M-Estimation

Let us assume two random vectors  $X = (x_1, x_2, \dots, x_N)$  and  $Y = (y_1, y_2, \dots, y_N)$ . Eq. (3.35) defining cross-correntropy between these two vectors is a similarity measure because it is always positive and is maximum when the two vectors are the same. Moreover, it induces a distance function in the input space called the correntropy induced metric (CIM). A metric on a set  $\mathbb{N}$  is a function  $d : \mathbb{N} \times \mathbb{N} \rightarrow R$ . For all  $X, Y, Z$  in  $\mathbb{N}$ , a metric should satisfy the following properties.

1. Nonnegativity  $d(X, Y) \geq 0$ .
2. Identity  $d(X, Y) = 0$  if and only if  $X = Y$ .
3. Symmetry  $d(X, Y) = d(Y, X)$ .
4. Triangle inequality  $d(X, Z) \leq d(X, Y) + d(Y, Z)$ .

If Property 2 is dropped, a *pseudometric* is obtained. In the supervised learning case, a metric is a function of the error vector  $E = Y - X = [e_1, e_2, \dots, e_N]$  and induces a norm if it is translation-invariant and homogeneous, which are respectively defined as

$$d(X + a, Y + a) = d(X, Y) \quad (3.44)$$

$$d(\alpha X, \alpha Y) = |\alpha|d(X, Y) \quad (3.45)$$

**Definition 3.1.** For any two random vectors  $X = (x_1, x_2, \dots, x_N)$  and  $Y = (y_1, y_2, \dots, y_N)$  the correntropy induced metric is defined as

$$\text{CIM}(X, Y) = (v(0, 0) - v(X, Y))^{1/2}. \quad (3.46)$$

It can be easily proven that CIM satisfies the properties of nonnegativity, identity, symmetry, triangle inequality, and translation invariant, and thus is a well-defined metric.

1. Non-negativity.  $\text{CIM}(X, Y) \geq 0$  by realizing that correntropy is positive and bounded:  $0 < v(X, Y) \leq 1/\sqrt{2\pi}\sigma$ . It reaches its maximum if and only if  $X = Y$ .
2. Identity.  $\text{CIM}(X, Y) = 0$  if and only if  $X = Y$  by the same reasoning as 1.
3. Symmetry. It is easily verified by reversing the argument in the kernel.
4. Triangle inequality:  $\text{CIM}(X, Z) \leq \text{CIM}(X, Y) + \text{CIM}(Y, Z)$ . The proof is based on the kernel mapping and a vector construction in a feature space which is a well-defined Hilbert space. For  $X$  and  $Y$ , we construct two new vectors  $\tilde{X} = [\Phi(x_1), \dots, \Phi(x_N)]$  and  $\tilde{Y} = [\Phi(y_1), \dots, \Phi(y_N)]$  in the Hilbert space  $H_\kappa$ . The Euclidean distance  $\text{ED}(\tilde{X}, \tilde{Y})$  is

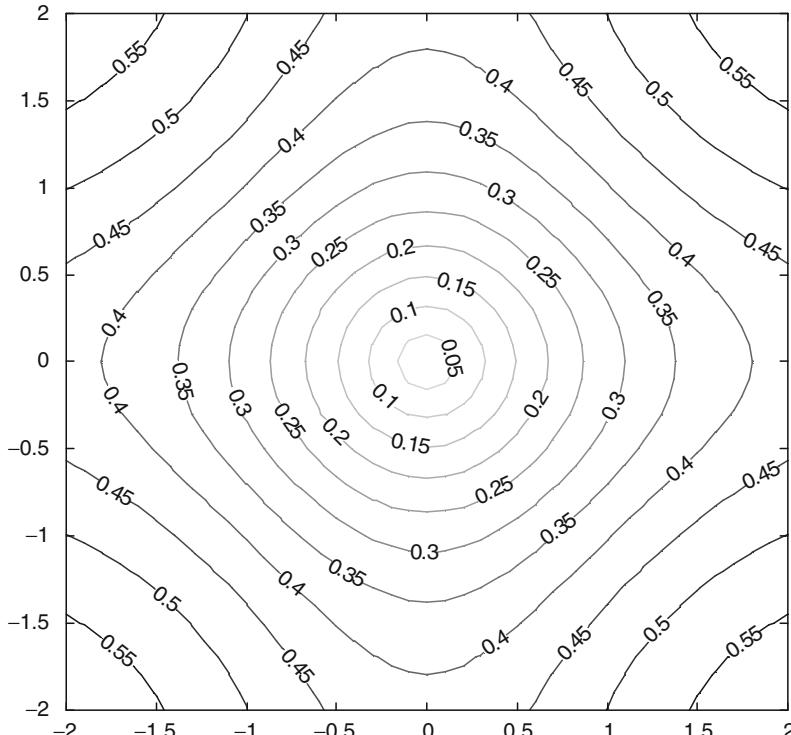
$$\begin{aligned}
ED(\tilde{X}, \tilde{Y}) &= \sqrt{<(\tilde{X} - \tilde{Y}), (\tilde{X} - \tilde{Y})>} \\
&= \sqrt{<\tilde{X}, \tilde{X}> - 2 <\tilde{X}, \tilde{Y}> + <\tilde{Y}, \tilde{Y}>} \\
&= \left( \sum_{i=1}^N G_\sigma(x_i - x_i) - 2 \sum_{i=1}^N G_\sigma(x_i - y_i) + \sum_{i=1}^N G_\sigma(y_i - y_i) \right)^{1/2} \\
&= \sqrt{2N(G_\sigma(0) - v(X, Y))} = \sqrt{2N} CIM(X, Y).
\end{aligned} \tag{3.47}$$

Therefore

$$CIM(X, Z) = \frac{ED(\tilde{X}, \tilde{Z})}{\sqrt{2N}} \leq \frac{ED(\tilde{X}, \tilde{Y})}{\sqrt{2N}} + \frac{ED(\tilde{Z}, \tilde{Y})}{\sqrt{2N}} = CIM(X, Y) + CIM(Y, Z). \tag{3.48}$$

This completes the proof.

It can also be shown that this metric is translation invariant for translation-invariant kernels such as the Gaussian kernel, so we can denote  $CIM(X, Y)$  as  $CIM(Y - X)$ . However, CIM is not homogeneous so it cannot further induce a norm on the sample space. The contour plots of constant CIM between two samples, one at the origin in 2D space, are shown in Figure 3.9.



**Fig. 3.9.** Contours of  $CIM(X, 0)$  in 2D sample space (kernel size is set to 1) (from [201]).

When the ECC is used as a cost function to train adaptive systems, we actually make the system output close to the desired signal in the CIM sense. Figure 3-9, which depicts the contour lines of equal CIM, shows that when the error vector is close to zero, CIM is equivalent to the  $L_2$ -distance (circular contours); when the error gets larger, CIM becomes an  $L_1$ -distance (diamond contour); and eventually when the error is large the metric levels off and becomes insensitive to the value of the error vector (approaching  $L_0$ ; i.e., a counting norm). The concepts of large and small here are related to the kernel size utilized in the Gaussian function. This intuitively explains the robustness of ECC, but we can be more precise by putting ECC in the general framework of M-estimation [201].

**Theorem 3.4.** *The error entropy criterion implemented with kernels is a robust cost function (in the sense of Huber's robust statistics).*

*Proof.* To show the relationship between ECC and M-estimation the error weighting function in Eq. (3.40) becomes  $(\text{CIM}(e))^2$ ; that is,

$$\rho(e) = (1 - \exp(-e^2/2\sigma^2))/\sqrt{2\pi}\sigma. \quad (3.49)$$

It is easy to see that this  $\rho(e)$  obeys the properties of Eq. (3.41). Therefore, according to Huber's theory of robust estimation, ECC as defined in Eq. (3.35) is equivalent to the following M-estimation problem,

$$\min_w \sum_{i=1}^N \rho(e_i) \quad (3.50)$$

or the following weighted least square problem,

$$\min_w \sum_{i=1}^N \gamma_i e_i^2. \quad (3.51)$$

The weight function  $\gamma(e)$  is defined by

$$\gamma(e) = \rho'(e)/e, \quad (3.52)$$

where  $\rho'$  is the derivative of  $\rho$ . Therefore

$$\gamma(e) = \exp(-e^2/2\sigma^2)/\sqrt{2\pi}\sigma^3. \quad (3.53)$$

This means that large errors get larger attenuation, thus the estimation is resistant to outliers. The weighting function of Eq. (3.53) is very similar to the bi-square method in Table 3.1 where  $\alpha$  is a tuning constant.

It turns out that the square of the Taylor expansion of Eq. (3.53) to the first order is the weighting function of bi-square and the kernel size  $\sigma$  substitutes the tuning constant in bi-square. Notice that in ECC the kernel size has a double role of representing the error PDF well and at the same time attenuating outliers, which sometimes may be too constraining when the number of samples is small. But in general it shows that the cost function has a built-in robustness because of the local estimation produced by the kernel.

### 3.8 Normalized Information Potential as a Pseudometric

In this section, we show the relation between the error entropy criterion of Eq. (3.36) and the error correntropy criterion, and therefore also the relation between EEC and M-estimation. Let us look at the EEC criterion in a way conducive to a similarity measure interpretation. Take the first-order difference between elements of  $Z$  and denote it by  $\Delta z_{ij} = z_i - z_j$ , likewise for  $Y$ , and construct vectors

$$\begin{aligned}\Delta Z &= (\Delta z_{11}, \Delta z_{12}, \dots, \Delta z_{21}, \Delta z_{22}, \dots, \Delta z_{NN}) \\ \Delta Y &= (\Delta y_{11}, \Delta y_{12}, \dots, \Delta y_{21}, \Delta y_{22}, \dots, \Delta y_{NN}).\end{aligned}$$

The correntropy between these two new vectors is

$$\hat{v}(\Delta Z, \Delta Y) = \frac{1}{N^2} \sum_{j=1}^N \sum_{i=1}^N G_\sigma(\Delta z_{ij} - \Delta y_{ij}). \quad (3.54)$$

Because  $\Delta z_{ij} - \Delta y_{ij} = (z_i - z_j) - (y_i - y_j) = e_i - e_j$  we obtain

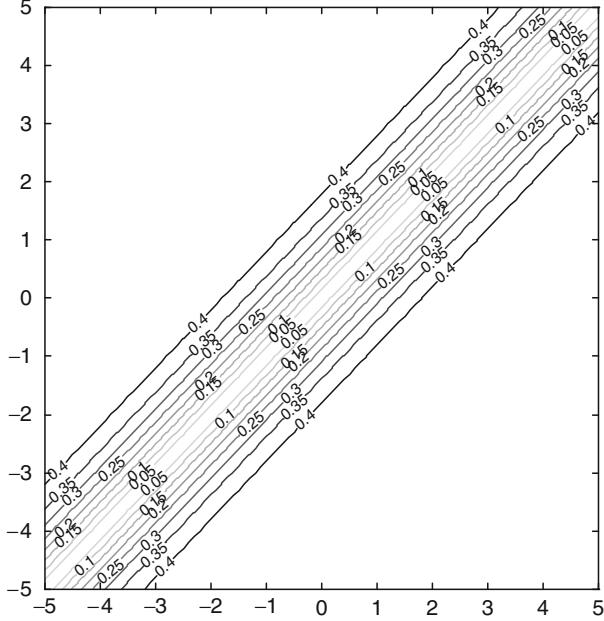
$$\hat{v}(\Delta Z, \Delta Y) = \frac{1}{N^2} \sum_{j=1}^N \sum_{i=1}^N G_\sigma(e_i - e_j) = \hat{V}(E). \quad (3.55)$$

Correntropy is a similarity measure, therefore so is the error entropy. The former compares the components of  $Z$  and  $Y$  directly whereas the error IP compares their first-order differences. Using this argument, it is easy to show that a simple normalization of the information potential is a pseudometric. Define the function  $V_N(Z, Y) = \sqrt{(V(0) - V(E))/V(0)}$ . By substituting Eq. (3.55) and noting that  $v(0, 0) = V(0)$  we get

$$V_N(Z, Y) = \left( \frac{v(0, 0) - v(\Delta Z, \Delta Y)}{V(0)} \right)^{1/2} = \frac{CIM(\Delta Z, \Delta Y)}{V(0)} \quad (3.56)$$

The normalized IP loses the property of identity when one takes the first order difference between variables. In other words, with  $\Delta Z$  available one can only recover the original  $Z$  up to an undetermined shifting constant (the mean of  $Z$ ). Let us calculate the MSE between  $\Delta Z$  and  $\Delta Y$  to gain some more insights. After some manipulations, we obtain  $MSE(\Delta Z, \Delta Y) = 2\text{var}(e)$ , which tells us two things: first the error IP of Eq. (3.55) is equivalent to the error variance when the kernel size is large; second it is not sensitive to the error mean which coincides with the previous analysis of MEE. Figure 3.10 shows the contour plots of  $VM(X, 0)$  to compare with the CIM contours of Figure 3.9.

The EEC criterion actually minimizes the distance between  $Z$  and  $Y$  in the IP sense or between  $\Delta Z$  and  $\Delta Y$  in the CIM sense. In fact, if we define the new error variable  $\Delta e_{ij} = \Delta z_{ij} - \Delta y_{ij} = e_i - e_j$  and assume that  $e$  is the actual random variation contained in the data, EEC is equivalent to the following M-estimation problem



**Fig. 3.10.** Contours of  $VM(X, \theta)$  in 2D sample space. It is a pseudometric (kernel size is set to 1) (from [200]).

$$\min \sum_{j=1}^N \sum_{i=1}^N \rho(\Delta e_{ij}), \quad (3.57)$$

or the weighted least square problem

$$\min \sum_{j=1}^N \sum_{i=1}^N \gamma(e_{ij})(\Delta e_{ij})^2 \quad (3.58)$$

with  $\rho(\cdot)$  and  $\gamma(\cdot)$  defined in Eqs. (3.49) and (3.53). Therefore, it becomes clear that the EEC criterion is also a method of M-estimation which further explains why EEC is a cost function that is resistant to outliers.

The basic assumption about errors in regression is they are purely stochastic white and follow a Gaussian distribution with zero mean and constant variance  $\sigma^2$ . Then the new error variable  $\Delta e_{ij}$  also follows a Gaussian distribution with zero mean and constant variance  $2\sigma^2$  for  $i \neq j$ . In the case of  $i = j$ , it equals zero and has no effect on the estimation. On the other hand, a small deviation from the above error model assumption (i.e., outliers present in the errors) corresponds to a small deviation of our new error random variable from the Gaussian model. Suppose the errors are drawn from the following PDF (Middleton model [220])

$$p_E(e) = (1 - \varepsilon) \times N(0, \sigma^2) + \varepsilon \times N(m_2, \sigma_2^2). \quad (3.59)$$

Here,  $\varepsilon$  denotes the degree of deviation which is usually much smaller than 1 and  $\sigma_2 \gg \sigma$ . Because  $e_i, e_j$  ( $i \neq j$ ) are independent, the PDF of  $\Delta e_{ij}$  is

$$p_E(de) = (1 - \varepsilon)^2 N(0, 2\sigma^2) + \varepsilon(1 - \varepsilon)N(m_2, \sigma_2^2 + \sigma^2) + \varepsilon(1 - \varepsilon) \\ \times N(-m_2, \sigma_2^2 + \sigma^2) + \varepsilon^2 N(0, 2\sigma_2^2). \quad (3.60)$$

However, by introducing the difference operation, the noise power is almost doubled (recall that  $MSE(\Delta Z, \Delta Y) = 2\text{var}(e)$ ) and the outlier effect is also double with respect to ECC. To summarize, let us compare ECC, EEC, and MSE by rewriting their definitions

$$\max v(Z, Y) = \int_e G_\sigma(e)p_E(e)de \\ \max V(E) = \int_e p_E(e)p_E(e)de \\ \min MSE(E) = \int_e e^2 p_E(e)de.$$

The learning process is nothing but the evolution of the error PDF according to a specific criterion. The problem of ECC and other M-estimation methods such as bi-squares is that one has to carefully anneal the kernel size or the tuning constant to balance efficiency with robustness. Intuitively, ECC is a local criterion because it only cares about the local part of the error PDF falling within the kernel bandwidth. When the error modes are far from the origin, they fall outside the kernel bandwidth, and the learning is insensitive to them. When one weights the error PDF by itself as in EEC, the error modes are easily detected anywhere in the error space with the advantage of data efficiency, but the kernel size still needs to be properly annealed as we saw early in this chapter. Whereas ECC is robust to outliers because it is local, EEC achieves robustness and efficiency by selfadjusting the localness of the weighting function based on the error distribution. The only problem with EEC is how to determine the location of the error PDF because the criterion is shift-invariant. Practically this is achieved by biasing the system output to the desired signal mean to make the error mean equal to zero. However, when the error PDF is nonsymmetric or has heavy tails the estimation of the mean error is problematic.

Finally, since we are utilizing gradient descent procedures, the adaptation of EEC and ECC are much simpler than the recursive approaches used in M-estimation.

### Centered Error Entropy Criterion

Our goal of supervised training is to make most of the errors equal to zero, therefore we can construct naturally an augmented criterion so that it minimizes the error entropy with respect to 0. A natural idea to automatically

achieve this goal is to locate the peak of the error PDF at the origin. Fixing the error peak at the origin in EEC is obviously better than the conventional method of shifting the error based on the mean of the desired response. Denote  $e_0 = 0$ ,  $E_C = [e_0, E]$  where  $E$  is the error vector produced by the adaptive system and  $e_0$  serves as a point of reference, which we call here a fiducial point. Then the error IP with a fiducial point at the origin becomes

$$V(E_C) = \frac{1}{(N+1)^2} \sum_{j=0}^N \sum_{i=0}^N \kappa_\sigma(e_i - e_j) = \frac{1}{(N+1)^2} \left[ 2 \sum_{i=1}^N \kappa_\sigma(e_i) + \sum_{j=1}^N \sum_{i=1}^N \kappa_\sigma(e_i - e_j) + \kappa_\sigma(0) \right]. \quad (3.61)$$

The above cost function is nothing but a weighted combination of ECC and EEC. According to our understanding, the EEC term minimizes the error entropy and the ECC term anchors the main peak of the error PDF at the origin which fulfills the design goal. In general, we can write the *centered error entropy criterion* (CEEC) as

$$J = \lambda \sum_{i=1}^N \kappa_\sigma(e_i) + (1-\lambda) \sum_{j=1}^N \sum_{i=1}^N \kappa_\sigma(e_i - e_j), \quad (3.62)$$

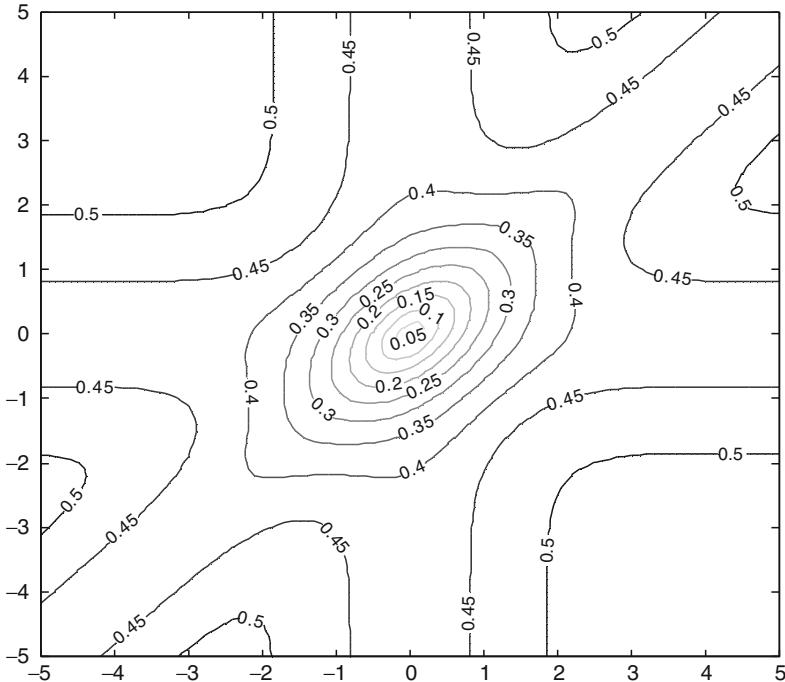
where  $\lambda$  is a weighting constant between 0 and 1 that can be interpreted by how many fiducial “samples” one puts at the origin. The more samples one puts at the origin, the larger  $\lambda$  becomes. When  $\lambda = 0$ , Eq. (3.62) reduces to EEC; when  $\lambda = 1$ , it is ECC, and  $\lambda = 1/3$  gives Eq. (3.61).

The advantage of CEEC versus the practical procedure outlined in Section 3.3 is that Eq. (3.62) automatically locates the main peak of the error PDF and fixes it at the origin even in the cases where the estimation of the error mean is not robust (i.e., the error PDF is not symmetric or has heavy tails). More significantly, we also can show that the CEEC actually induces a well-defined metric. We use Eq. (3.61) to show that it induces a metric as correntropy and an interesting observation is that it becomes sensitive to direction because it contains the MEE term (Figure 3.11). Most metrics are symmetric in the following sense

$$SM(\dots, e_i, \dots) = SM(\dots, -e_i, \dots). \quad (3.63)$$

However, CEEC favors errors with the same sign and penalizes the directions where errors have different signs. If the errors are small with respect to the kernel size, this metric can be described by the linear term of Eq. (3.61). Assume  $N = 2$  and by simple manipulations, we have

$$r^2 \propto \begin{pmatrix} e_1 \\ e_2 \end{pmatrix}^T \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}^{-1} \begin{pmatrix} e_1 \\ e_2 \end{pmatrix}. \quad (3.64)$$



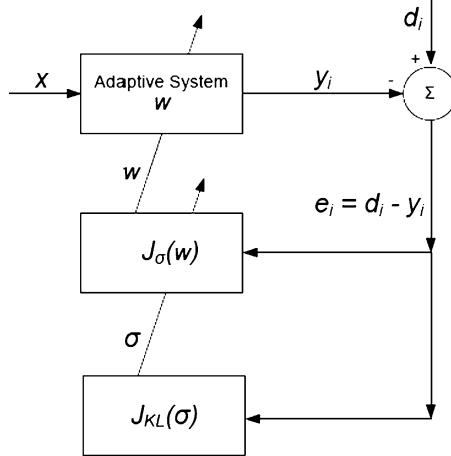
**Fig. 3.11.** Contours of the metric induced by CEEC in 2D sample space (scaling and rotating variant). It penalizes the directions of  $3\pi/4$  and  $7\pi/4$  where the errors have different signs (kernel size is 1)(from [200]).

This is the squared Mahalanobis distance from  $e$  to 0 with covariance matrix  $\Sigma = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$ .

The eigenvectors are  $(0.707, 0.707)$ ,  $(-0.707, 0.707)$  with the corresponding eigenvalues 3, 1. This justifies the elliptic pattern in Figure 3.11, and it may find practical applications.

### 3.9 Adaptation of the Kernel Size in Adaptive Filtering

The kernel size in EEC and ECC can be considered one more parameter to be adapted during learning, unlike the work in density estimation where a reasonable cost function for selecting the kernel size remains elusive. This shows that filtering is a simpler problem than density estimation, but a thorough investigation of the best cost to adapt the kernel size parameter is still lacking. Figure 3.12 shows the overall block diagram for the adaptation of EEC and ECC when the kernel size parameter is also being adapted. Notice that now



**Fig. 3.12.** Composite adaptation of filter parameters and the kernel size.

we have two cost functions, one to adapt the parameters of the filter  $J_\sigma(w)$ , and a second cost function  $J_{KL}(\sigma)$  to adapt the kernel size in the estimation of the EEC or ECC costs.

The optimality condition proposed to adapt the kernel size is derived from nonparameteric regression concepts [133], where the kernel size is understood as a compromise between bias and variance of the regressor. As such we propose the use of the Kullback-Leibler divergence between the true error PDF and the estimated PDF as our criterion to adapt the kernel size parameter online [302].

Let  $\hat{p}_\sigma(e)$  be the estimated density from a window of  $N$  samples of the error, evaluated using the Gaussian kernel with width  $\sigma : \hat{p}_\sigma(e) = 1/N \sum_{i=1}^N G_\sigma(e - e_i)$ . From an information-theoretic perspective, an optimal value of  $\sigma$  would be one that minimizes the discriminant information between the estimated density and the true density  $p(e)$  of the errors. Therefore, the cost function for optimizing the kernel size is:

$$D_{KL}(p||\hat{p}_\sigma) = \int p(e) \log \left( \frac{p(e)}{\hat{p}_\sigma(e)} \right) de = \int p(e) \log p(e) de - \int p(e) \log \hat{p}_\sigma(e) de. \quad (3.65)$$

The first term in Eq. (3.65) is independent of the kernel size. Therefore, minimizing  $D_{KL}(p||\hat{p}_\sigma)$  with respect to  $\sigma$  is equivalent to maximizing the second term of Eq. (3.65), which is nothing but the cross-entropy between the true PDF and the estimated PDF with kernels. Therefore our cost function to optimize for the value of  $\sigma$  is simply

$$J_{KL}(\sigma) = E[\log(\hat{p}_\sigma(e)] \quad (3.66)$$

which can also be interpreted as a maximum log-likelihood condition. Using the sample estimator for the expectation operator, we obtain

$$\hat{J}_{KL}(\sigma) = \frac{1}{N} \sum_{i=1}^N \log \left( \frac{1}{N} \sum_{j=1}^N G(e_i - e_j) \right). \quad (3.67)$$

Taking the derivative of  $J_{KL}(\sigma)$  with respect to  $\sigma$  yields

$$\frac{\partial J_{KL}(\sigma)}{\partial \sigma} = E \left[ \frac{\sum_{i=1}^N \exp\left(\frac{-(e - e_i)^2}{2\sigma^2}\right) \left(\frac{(e - e_i)^2}{\sigma^3} - \frac{1}{\sigma}\right)}{\sum_{i=1}^N \exp\left(\frac{-(e - e_i)^2}{2\sigma^2}\right)} \right]. \quad (3.68)$$

Going back to our original adaptive filter configuration, using the above equation and adapting it to an online application we can formulate a simple gradient ascent search rule to update the kernel size, at every weight update step of the adaptive filter as

$$\begin{aligned} \sigma(n+1) &= \sigma(n) + \eta \frac{\partial J_{KL}(\sigma(n))}{\partial \sigma(n)} \\ \sigma(n+1) &= \sigma(n) + \eta E \left[ \frac{\sum_{i=n-L}^{n-1} \exp\left(\frac{-(e - e(i))^2}{2\sigma^2(n)}\right) \left(\frac{(e - e(i))^2}{\sigma^3(n)} - \frac{1}{\sigma(n)}\right)}{\sum_{i=n-L}^{n-1} \exp\left(\frac{-(e - e(i))^2}{2\sigma^2(n)}\right)} \right], \end{aligned} \quad (3.69)$$

where  $n$  is the current time index, and  $L$  is the window size to estimate  $J$ . We can use a stochastic approximation of the gradient by dropping the expectation operator and evaluating the operand at the current sample of the error. Therefore the final update rule becomes:

$$\sigma(n+1) = \sigma(n) + \eta \left[ \frac{\sum_{i=n-L}^{n-1} \exp\left(\frac{-(e(n) - e(i))^2}{2\sigma^2(n)}\right) \left(\frac{(e(n) - e(i))^2}{\sigma^3(n)} - \frac{1}{\sigma(n)}\right)}{\sum_{i=n-L}^{n-1} \exp\left(\frac{-(e(n) - e(i))^2}{2\sigma^2(n)}\right)} \right]. \quad (3.70)$$

The computational complexity of this adaptation technique is  $O(L)$ , where  $L$  is the length of the window used for computing the density estimate. A special case of the above update rule arises when we take  $L = 1$ , in which case Eq. (3.70) takes the simple form of a stochastic gradient update to adapt the kernel size as

$$\sigma(n+1) = \sigma(n) + \eta \left( \frac{(e(n) - e(n-1))^2}{\sigma^3(n)} - \frac{1}{\sigma(n)} \right). \quad (3.71)$$

Equation (3.68) deserves a closer examination, as it provides useful insights about the steady-state values of  $\sigma$ . Clearly, the condition for convergence of  $\sigma$  is when

$$\sum_{i=n-L}^{n-1} \exp\left(\frac{-(e(n) - e(i))^2}{2\sigma^2(n)}\right) \left(\frac{(e(n) - e(i))^2}{\sigma^3(n)} - \frac{1}{\sigma(n)}\right) = 0. \quad (3.72)$$

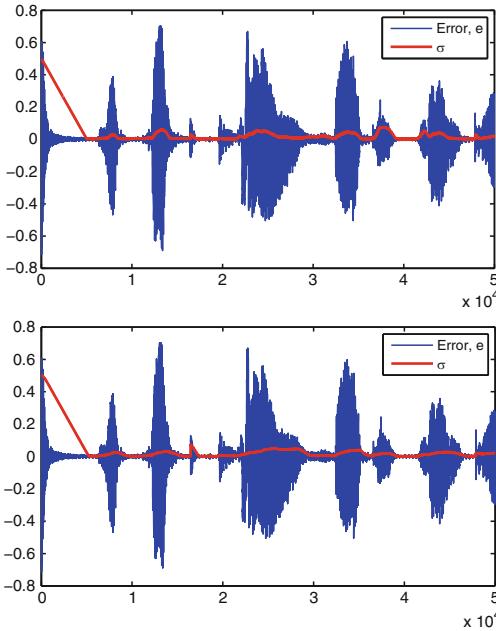
A little rearrangement yields

$$\sigma^2(n) = \frac{\sum_{i=n-L}^{n-1} \exp\left(\frac{-(e(n) - e(i))^2}{2\sigma^2(n)}\right) ((e(n) - e(i))^2)}{\sum_{i=n-L}^{n-1} \exp\left(\frac{-(e(n) - e(i))^2}{2\sigma^2(n)}\right)}. \quad (3.73)$$

It is interesting to observe that the right-hand side of Eq. (3.73) is in fact the computation of a weighted variance of the errors around the current sample  $e(n)$ . The exponential weighting given to the squared differences  $(e(n) - e(i))^2$  provides a windowing or a localization to the variance computation. The localization is not in time, but in the space of the magnitude of the errors. This means that values close to the current sample  $e(n)$  receive higher weight while computing the variance. It can therefore be seen that using the adaptation rule of Eq. (3.70), the kernel width converges to the value of the localized variance of the error. This property is particularly useful when the distribution of errors is multimodal.

Although the stochastic gradient approximation of Eq. (3.70) reduces computational complexity as compared to Eq. (3.69), it is more easily affected by outliers in the data. In practice, this effect can be alleviated by having a small regularization constant  $\varpi$  added to the denominator of Eq. (3.71). The initial condition for the kernel size should be much larger than the one expected from density estimation considerations, because for large kernel sizes the cost function becomes closer to the quadratic cost as discussed above.

We have tested this method in adaptive noise cancellation of speech signals masked by heavy noise, and the kernel size adjusts itself to match the changes in variance of the error. Figure 3.13 shows examples of the kernel size through adaptation as a function of window size. This figure corroborates the analysis that the kernel size estimates the local variance of the error signal, which is a good thing because the error power changes across iterations in an unknown way. But of course,  $J_{KL}(\sigma)$  is still not linked to the performance of the adaptation algorithm, and moreover, it depends upon the error distribution, which is beyond the control of the designer. Experience shows that the values obtained are reasonable and that the adaptation is faster than with constant kernel size.



**Fig. 3.13.** Kernel size being adapted online (top:  $L = 1$ ; bottom  $L = 100$ ) (from [302]).

### 3.10 Conclusions

The purpose of Chapter 3 is to go beyond mean square error adaptation by using the argument of the logarithm of Renyi's quadratic entropy. Because the logarithm does not change the location of the optimum in parameter space because it is a monotonic function, the adaptation algorithms are simplified when the IP is used as the cost instead of Renyi's entropy directly.

When the information potential estimator is utilized, the PDF is never explicitly estimated, so the cost works with pairs of samples of the error distribution, which can be thought of as a nonlinear projection of the error into a high-dimensional space. Therefore, although the filter is linear, the solution obtained by the MEE algorithms does not necessarily coincide with the MSE solution. It really depends on the error distribution: in cases where the error PDF is approximately Gaussian, the two solutions provide similar results, but in other cases, the solutions will differ.

We presented the basic block mode gradient descent update for the MEE cost function, studied some of its properties, and addressed the features of the EEC performance surface to get a better feel for the adaptation. We found that EEC tends to have a peaky performance surface, with exponential tails, with potential local minima in the tails region. Therefore in most of the space the search using gradient descent can be expected to be slow. However, near the optimum the steepness of the performance curve is faster than quadratic.

We have also shown that there is always a neighborhood around the minimum, controlled by the kernel size, where a quadratic approximation of the performance surface is possible. This simplifies the treatment of the adaptation near the optimum, but still leaves open the problem of finding fast algorithms overall.

This chapter also provided a clear statistical reason why the error entropy criterion (and its MEE algorithm) is robust to outliers by establishing the link between entropy cost functions and robust statistics. This link was accomplished through a new function called correntropy that is further studied in Chapter 10. But the correntropy error criterion (and its MCC learning algorithm) is rather interesting because it creates a new metric in the space of the errors that has unique properties. In fact, unlike normal metrics it uses different evaluations of distance depending upon how far the points of interest are; when the samples are close the metric is  $L_2$ , farther apart it becomes  $L_1$ , and farther out it saturates to  $L_0$ . The values of near and far are relative to the kernel size utilized in the Gaussian function, which means that a single constant has a tremendous effect in the evaluation of similarity. When placed in the framework of M-estimation, we showed that the derivative of (CIM)<sup>2</sup> is effectively the weighting function in weighted least squares. The conventional threshold in M-estimation is replaced by the kernel size. The comparisons show that the ECC is comparable to M-estimation, but it can be easily integrated online as a cost function because it is continuous, unlike Huber's (Table 3.1).

The chapter also linked ECC with EEC, by showing that the ECC of the error differential yields EEC, which shows that the noise robustness of the MEE algorithm can also be interpreted in terms of weighted least squares. This close relation between the two criteria suggests a new criterion that automatically centers the errors, unlike EEC, which we called the centered EEC and which is based on a metricTherefore it can be used as a criterion without the requirement of centering the errors. As a conclusion we see that the MEE and MCC algorithms can be used with advantage when the noise PDF has a long tail. This condition happens even when the model system that creates our data is linear, but the observation noise has outliers.

The final topic covered in this chapter deals with the online adaptation of the kernel size, which in adaptive filtering makes a lot of sense because it can be considered as an extra parameter that needs to be adapted. However, the kernel size affects the cost function, so it requires a different cost function for adaptation. We have proposed a cost that minimizes the KL divergence between the true error PDF and the estimated error PDF. We showed that the method basically estimates local variance, it is easy to implement online, and the results are reasonable and follow the changes in error power that normally occur through adaptation. However, this cost is not related to the properties of the adaptation, which is a shortcoming; that is, we may want to adapt the kernel size to minimize the misadjustment or to increase the speed of adaptation. Therefore, the proposed method should be interpreted as just one of many possible ways of framing the kernel adaptation problem.

# Algorithms for Entropy and Correntropy Adaptation with Applications to Linear Systems

Deniz Erdogmus, Seungju Han, and Abhishek Singh

## 4.1 Introduction

This chapter develops several batch and online learning algorithms for the error entropy criterion (EEC) that are counterparts to the most widely used algorithms for the mean square error criterion (MSE). Because the chapter assumes knowledge of adaptive filter design, readers unfamiliar with this topic should seek a textbook such as [332] or [253] for a review of fundamentals. But the treatment does not require an in-depth knowledge of this field. The case studies in this chapter address only adaptation of linear systems, not because entropic costs are particularly useful for the linear model, but because the solutions for linear systems are well understood and performance comparisons can be easily drawn. This chapter also considers applications of fast evaluations of the IP using the fast Gauss transform and incomplete Cholesky decomposition, and ends with an application of the error correntropy criterion (ECC) to adaptive noise cancellation.

Unlike the MSE criterion, there is no known closed form solution to find the optimal weights in EEC adaptation even in the finite impulse response (FIR) case. However, local search procedures based on the gradient can be used effectively as we demonstrated in Eq. (3.16) that implements a steepest descent procedure for  $V(e)$  estimated with the IP.

Table 4.1 shows the most widely used adaptive algorithms under the MSE criterion and the corresponding ones for EEC presented in this chapter. Because the goal of this chapter is to minimize EEC, we refer almost exclusively to the minimization of error entropy (MEE) algorithm in Eq. (3.16).

In the table,  $N$  is the number of samples,  $M$  the filter order, and  $L$  the number of samples in a window. Each of these algorithms was developed to answer a specific question [143]. The steepest descent has a smooth convergence to the minimum because it uses an average estimation of the gradient. The LMS is a stochastic approximation to the latter, and has the smallest

**Table 4.1.** Family of Algorithms

MSE Criterion	Error Entropy Criterion	Complexity MSE/EEC
Steepest descent	MEE (Minimum error entropy)	$O(N)/O(N^2)$
	MEE-RIP (recursive IP)	$-/O(L)$
LMS (least means squares)	MEE-SIG (stochastic gradient)	$O(M)/O(L)$
LMF (least mean fourth)	MEE-SAS (self adjusting stepsize)	$O(M)/O(N^2)$
NLMS (normalized LMS)	NMEE (normalized MEE)	$O(M)/O(N^2)$
RLS (recursive least squares)	MEE-FP (fixed point update)	$O(M^2)/O(M^2L)$

computational complexity of the group (two multiplications per weight), but the estimation of the gradient is noisy so the stepsize needs to be reduced about 1/10 when compared with the steepest descent. The LMF was introduced to speed up the convergence of the LMS, whereas the NLMS is very robust to changes in the input signal power. The RLS algorithm is the fastest of the group, approximating the Wiener solution at every sample, but it is far more computationally expensive than the others. The corresponding MEE algorithms serve a similar role, but are based on the error entropy cost function.

For the case of multiple system outputs, the cost function needs to be modified as the sum of marginal error entropies, or alternatively, the product of the individual information potentials. For a system with  $d$  outputs, the corresponding cost is

$$\min_w J(w) = \min_w \sum_{o=1}^d H_\alpha(e_o) \stackrel{\alpha \geq 1}{\equiv} \max_w \prod_{o=1}^d V_\alpha(e_o), \quad (4.1)$$

where  $e_o$  denotes the error signal for the  $o$ th output of the adaptive system. With this cost function, then the gradient has to be modified to

$$\frac{\partial J}{\partial w} = \sum_{o=1}^d \frac{1}{1-\alpha} \frac{\partial V_\alpha(e_o)}{\partial w} \text{ or } \sum_{o=1}^d \left( \prod_{p \neq o} V_\alpha(e_p) \right) \frac{\partial V_\alpha(e_o)}{\partial w}. \quad (4.2)$$

A second approach for multioutput situations is to minimize the joint entropy of the error vector, however, in general this approach requires an exponentially increasing number of samples, so the product approach is preferred in practice.

## 4.2 Recursive Information Potential for MEE (MEE-RIP)

The MEE algorithm already presented in Section 3.3 is  $O(N^2)$  in the number of samples due to the double summation in Eq. (3.16). For online scenarios, we notice that it is possible to obtain a recursive formula to update the

information potential estimate when a new sample is acquired. Suppose that at time  $n$ , when we already have  $n$  samples, the quadratic information potential is estimated to be

$$\hat{V}_n(X) = \frac{1}{n^2} \sum_{j=1}^n \sum_{i=1}^n \kappa_\sigma(x_j - x_i). \quad (4.3)$$

Suppose at time  $n + 1$  we get a new sample  $x_{n+1}$  and we wish to update our estimate. Assuming that the kernel function is selected to be an even-symmetric PDF,

$$\begin{aligned} \hat{V}_{n+1}(X) &= \frac{1}{(n+1)^2} \sum_{j=1}^{n+1} \sum_{i=1}^{n+1} \kappa_\sigma(x_j - x_i) \\ &= \frac{n^2}{(n+1)^2} \hat{V}_n + \frac{1}{(n+1)^2} \left[ 2 \sum_{i=1}^n \kappa_\sigma(x_{n+1} - x_i) + \kappa_\sigma(0) \right]. \end{aligned} \quad (4.4)$$

Once the information potential estimate is updated, the new entropy estimate can be obtained by simply evaluating  $\hat{H}_{n+1}(X) = -\log \hat{V}_{n+1}(X)$ . This recursion yields exactly the same estimate as the batch estimator in Eq. (4.3) at every time instance, therefore we call this the exact recursive entropy estimator. The beauty of Eq. (4.4) is that it is online, and of complexity  $O(n)$  instead of  $O(n^2)$ , but  $n$  is increasing with iterations. This exact recursion is useful for estimating the entropy of stationary signals, however, due to its increasing memory depth, it is not suitable for nonstationary environments. Therefore, we employ the fixed forgetting factor approach to derive one estimator that would serve satisfactorily in such situations.

### Recursive Estimator with Forgetting Factor

We start by defining a recursive Parzen window estimate. Suppose that at time  $n$ , the PDF estimate  $p_n(x)$  for  $p_X(x)$  is available. Using the new sample  $x_{n+1}$ , we update this PDF estimate according to

$$p_{n+1}(x) = (1 - \lambda)p_n(x) + \lambda\kappa_\sigma(x - x_{n+1}). \quad (4.5)$$

The initial PDF estimate could be selected as  $p_1(x) = \kappa_\sigma(x - x_1)$ . Substituting the recursive PDF estimate in Eq. (4.3) for the actual PDF in the definition given in Eq. (4.5), we obtain the recursion for the quadratic information potential.

$$\begin{aligned} \bar{V}_{n+1} &= Ex[p_{n+1}(X)] = (1 - \lambda)Ex[p_n(X)] + \lambda Ex[\kappa_\sigma(X - x_{n+1})] \\ &\cong (1 - \lambda)\bar{V}_n + \frac{\lambda}{L} \sum_{i=n-L+1}^n \kappa_\sigma(x_i - x_{n+1}). \end{aligned} \quad (4.6)$$

The recursion in Eq. (4.6) is named the *forgetting recursive entropy estimator*. The parameters  $\lambda$ ,  $L$ , and  $\sigma$  are called the *forgetting factor*, *window length*, and *kernel size*, respectively. These free design parameters have an effect on the convergence properties of this recursive entropy estimator, and have been fully investigated in [90].

An interesting relationship between the exact and forgetting recursive entropy estimators of Eqs. (4.4) and (4.6) is that, if we replace the fixed memory depth and the fixed window length of Eq. (4.6) with dynamic ones, the two recursions asymptotically converge to the same value. In order to see this, we set  $\lambda = 1 - n^2/(n+1)^2$  and  $L = n$ . Then taking the limit of the difference between Eq. (4.4) and Eq. (4.6) as  $n$  goes to infinity yields

$$\begin{aligned} & \lim_{n \rightarrow \infty} (\hat{V}_{n+1} - \bar{V}_{n+1}) \\ &= \lim_{n \rightarrow \infty} \left[ \frac{n^2}{(n+1)^2} \hat{V}_n - (1-\lambda) \bar{V}_n + \frac{1}{(n+1)^2} \kappa_\sigma(0) \right. \\ & \quad \left. + \frac{2}{(n+1)^2} \sum_{i=1}^n \kappa_\sigma(x_{n+1} - x_i) - \frac{\lambda}{n} \sum_{i=1}^n \kappa_\sigma(x_i - x_{n+1}) \right] = 0. \end{aligned} \quad (4.7)$$

The important practical property of this recursive estimator is that it reduces the computational complexity from  $O(n^2)$  to  $O(L)$ . This is a drastic reduction in the computational requirements. The forgetting recursive entropy estimator also enjoys a reduced memory requirement compared to the exact recursion and the batch formula which is very useful to track changes of the input in locally stationary signals.

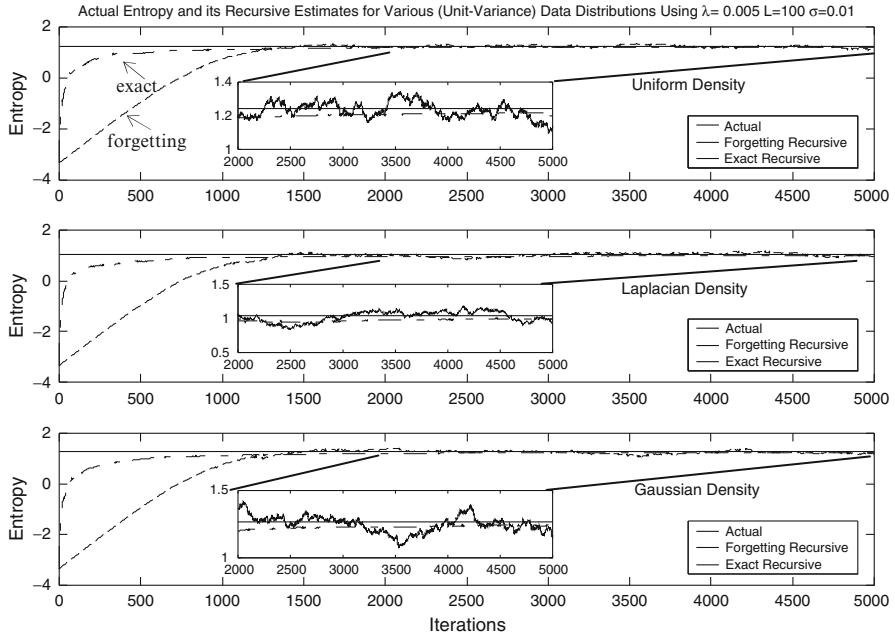
### Case Study for MEE-RIP

In this section, we investigate the performance of the recursive entropy estimators proposed above. We start by demonstrating the convergence properties of both estimators to the true entropy value of the PDF underlying the data that are being presented. In these simulations, we have utilized 5000 samples generated by zero-mean, unit-variance uniform, Laplacian, and Gaussian distributions. For these density functions, both the exact and forgetting recursions are evaluated and displayed in Figure 4.1.

The Gaussian kernel has size  $\sigma = 0.01$ , the window length is 100 samples and the forgetting factor  $\lambda = 0.05$ . Several simulations were run on these data to investigate the effect of the forgetting factor, window length, and kernel size in the estimates. They are summarized in the Tables 4.2 through 4.4.

In recursive estimates there is an intrinsic trade-off between speed and variance, which the designer must consider in selecting the forgetting factor (see Table 4.2).

As expected, the speed of convergence is not affected by the variations in the window size. However, the estimation variance after convergence is greatly affected because of the number of samples (Table 4.3). The trade-off in the



**Fig. 4.1.** Actual entropy and its exact and forgetting recursive estimates for uniform, Laplacian, and Gaussian densities (from [90]).

**Table 4.2.** Convergence for Different Forgetting Factors

$L = 100, \sigma = 0.01$	$\lambda = 0.001$	$\lambda = 0.003$	$\lambda = 0.01$
Convergence (samples)	8000	2500	1000
variance	$1.1 \times 10^{-4}$	$9.5 \times 10^{-4}$	$2.7 \times 10^{-3}$

**Table 4.3.** Effect of Window Length

$\lambda = 0.002, \sigma = 0.001$	$L = 10$	$L = 100$	$L = 1000$
Convergence (samples)	8000	2500	1000
variance	$6.7 \times 10^{-3}$	$1.7 \times 10^{-4}$	$2.2 \times 10^{-5}$

selection of this parameter is between the accuracy after convergence and the memory requirement.

Parzen windowing has a bias that increases with the kernel size, whereas its variance decreases with the kernel size. As expected, the smallest kernel size resulted in the largest variance and the largest kernel size resulted in the largest bias (Table 4.4).

**Table 4.4.** Effect of Kernel Size

$\lambda = .002, L = 001$	$\sigma = 0.001$	$\sigma = 0.01$	$\sigma = 0.1$	$\sigma = 1$
mean	$5 \times 10^{-2}$	$2.2 \times 10^{-2}$	$1.3 \times 10^{-2}$	$2.4 \times 10^{-1}$
variance	$3.9 \times 10^{-3}$	$1.6 \times 10^{-4}$	$2.9 \times 10^{-5}$	$3 \times 10^{-5}$

### Gradient for the MEE-RIP

The previous experiments show the good properties of the recursive computation of the information potential, but for our purposes, we need the gradient of this quantity to train the FIR filter with MEE. Towards this goal we take the derivative of the recursive information potential with respect to the weights and write

$$\frac{\partial \bar{V}_n}{\partial w} = \frac{\partial \bar{V}_n}{\partial (e_i - e_n)} \frac{\partial (e_i - e_n)}{\partial w}. \quad (4.8)$$

Now using Eq. (4.5) for the first term on the right-hand side, we get

$$\frac{\partial \bar{V}_n}{\partial (e_i - e_n)} = (1 - \lambda) \frac{\partial \bar{V}_{n-1}}{\partial (e_i - e_n)} + \frac{\lambda}{L} \frac{\partial}{\partial (e_i - e_n)} \left( \sum_{i=n-L}^{n-1} G(e_i - e_n) \right) \quad (4.9)$$

and evaluating the derivative

$$\frac{\partial \bar{V}_n}{\partial (e_i - e_n)} = (1 - \lambda) \frac{\partial \bar{V}_{n-1}}{\partial (e_i - e_n)} + \frac{\lambda}{\sigma^2 L} \sum_{i=n-L}^{n-1} G(e_i - e_n)(e_i - e_n). \quad (4.10)$$

Notice that the first derivative on the right side of Eq. (4.9) is not zero, because  $V_{n-1}$  depends on the previous errors (it is called an ordered derivative). Now this term can be computed at each step by storing a single quantity  $\partial \bar{V}_{n-1} / \partial (e_n - e_i)$ , which is the gradient at the previous step, and computing the sum which is  $O(L)$  at each step. The initial condition is  $\partial \bar{V}_0 / \partial e_0 = 0$  and of course the sum is started with  $G(0 - e_1)(0 - e_1)$ . The gradient of the error with respect to the weights in Eq. (4.8) is  $\partial (e_i - e_n) / \partial w = x_n - x_i$  and it can be brought in this formulation easily to yield the gradient for the  $k$ th FIR weight as

$$\nabla_k \bar{V}(n) = (1 - \lambda) \frac{\partial \bar{V}_k(n-1)}{\partial (e_i - e_n)} + \frac{\lambda}{\sigma^2 L} \sum_{i=n-L}^{n-1} G(e_i - e_n)(e_i - e_n)(x_k(n) - x_k(i)). \quad (4.11)$$

This expression is the gradient of the MEE-RIP and substitutes Eq. (3.16), preserving the accuracy of the solution for appropriate  $\lambda$  and  $L$ .

### 4.3 Stochastic Information Gradient for MEE (MEE-SIG)

Having introduced the methodology to derive a recursive estimator for the information potential, we can still attempt to derive a stochastic information gradient using Widrow's intuition of dropping the expectation operator

in his famous LMS algorithm [332]. Both the recursive and the stochastic information gradient use online updates, but the strength of LMS lies in its ability to determine the optimal solution of the MSE criterion with extremely simple updates on the weight vector, which it computes using only the most recently acquired input signal. In this section, we derive a stochastic gradient for the minimum error entropy criterion, with the hope to obtain even a simpler estimator for the information potential. It is important to remember that the fundamental property that supports the stochastic update is an unbiased estimator of the gradient with the present sample.

Suppose that, in an online adaptation scenario, we approximate the  $\alpha$ -information potential stochastically by the argument of the expectation operation. Therefore, dropping  $E[\cdot]$  in  $V_\alpha(e) = E[p_e^{\alpha-1}(e)]$  and substituting the required PDF by its Parzen estimate over the most recent  $L$  samples at time  $n$ , the stochastic information potential estimate becomes

$$\hat{V}_\alpha(e(n)) \approx \left( \frac{1}{L} \sum_{i=n-L}^{n-1} \kappa_\sigma(e_n - e_i) \right)^{\alpha-1}. \quad (4.12)$$

If we substitute in Eq. (3.18) we obtain

$$\begin{aligned} \frac{\partial \hat{V}_\alpha(e(n))}{\partial w_k} &= -\frac{(\alpha-1)}{L^{\alpha-1}} \left( \sum_{i=n-L}^{n-1} \kappa_\sigma(e_n - e_i) \right)^{\alpha-2} \\ &\quad \times \left[ \sum_{i=n-L}^{n-1} \kappa'_\sigma(e_n - e_i)(x_k(n) - x_k(i)) \right]. \end{aligned} \quad (4.13)$$

This is the general expression of the stochastic gradient of the IP for arbitrary  $\alpha$  and kernel (MEE-SIG). The first expression provides the weighting on the sample density translating the role of  $\alpha$  in Renyi's entropy. For  $\alpha = 2$  the first sum in Eq. (4.13) disappears as discussed in Chapter 2, and it provides a simpler expression. For the quadratic information potential using a Gaussian kernel, the SIG(L) gradient becomes

$$\frac{\partial \hat{V}_2(e(n))}{\partial w_k} = \frac{1}{\sigma^2 L} \left[ \sum_{i=n-L}^{n-1} G_\sigma(e_n - e_i)(e_n - e_i)(x_k(i) - x_k(n)) \right]. \quad (4.14)$$

Comparing with Eq. (3.18) the computation of this expression is proportional to the size of the time window used to estimate the information potential. Comparing with Eq. (4.11) SIG(L) corresponds to the MEE-RIP for  $\lambda = 1$ . In the extreme case,  $L = 1$  the MEE-SIG denoted by SIG(1) becomes,

$$\frac{\partial \hat{V}_\alpha(e(n))}{\partial w_k} = \frac{1}{\sigma^2} G_\sigma(e(n) - e(n-1))(e(n) - e(n-1))(x_k(n-1) - x_k(n)), \quad (4.15)$$

which can be interpreted as a weighted combination (by the Gaussian of the error difference) of the product of the incremental error and the incremental

input. SIG(1) is the EEC equivalent of the LMS algorithm, and the similarity between the two updates is easy to pinpoint. There are a lot of assumptions in this update, but the stochastic estimation literature [192] tells us that as long as the time estimator is unbiased, the average over iterations decreases its variance and converges to the same value. We only conducted experimental evaluations of the SIG(1), and the results show that it still finds extremes of the entropy cost in synthetic datasets (this is also demonstrated in Chapter 8).

### Training a Linear Combiner with SIG(L)

All the simulations presented in this chapter, unless otherwise specified, adapt a 2-tap FIR for system identification (Figure 3.1), and the desired response data are also generated by a 2-tap FIR with weight vector  $\mathbf{w}_* = [1, 2]^T$ . There is no measurement noise and the input to the filter is zero-mean, unit variance white Gaussian noise. This experimental setup is very simple but is used in this chapter because we can visualize the weight tracks during adaptation to evaluate convergence and we know that the stationary point is equal to the MSE cost because the system identification solution is unique. Figure 4.2 shows the effect of the kernel size ( $\sigma$ ) in the adaptation speed of the SIG ( $L = 100$ ) and  $\mu = 1$ . For a large range of kernel sizes the adaptation converges and the slowest convergence is achieved in intermediate values of the kernel size ( $\sigma = 0.5$ ). When the kernel size is smaller than required for

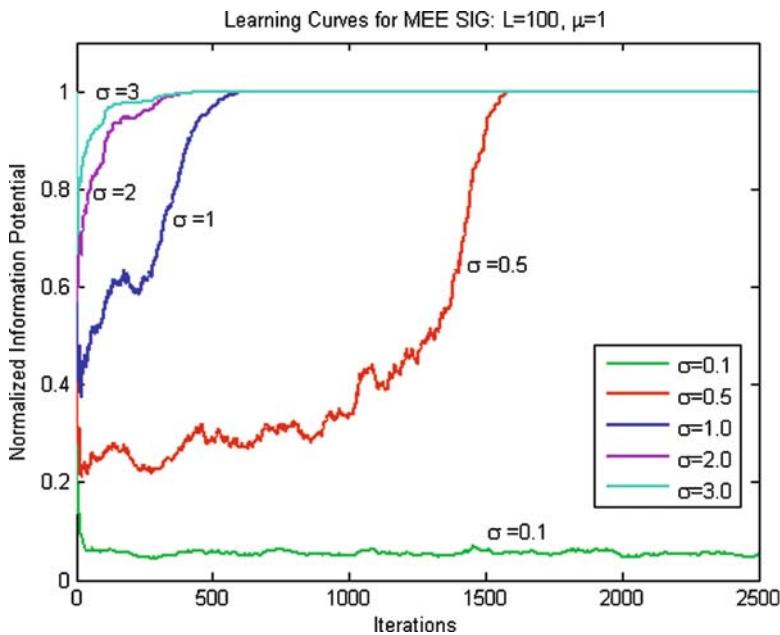
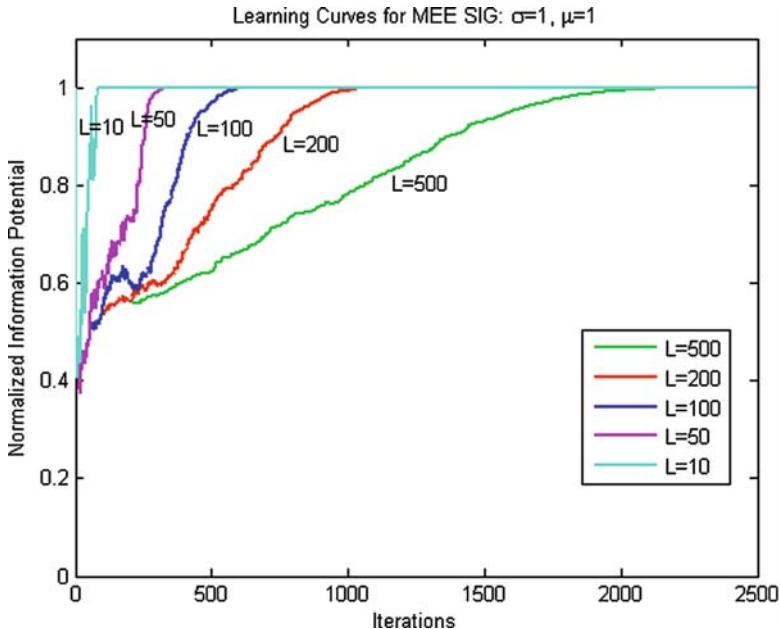


Fig. 4.2. The effect of kernel size in adaptation speed.



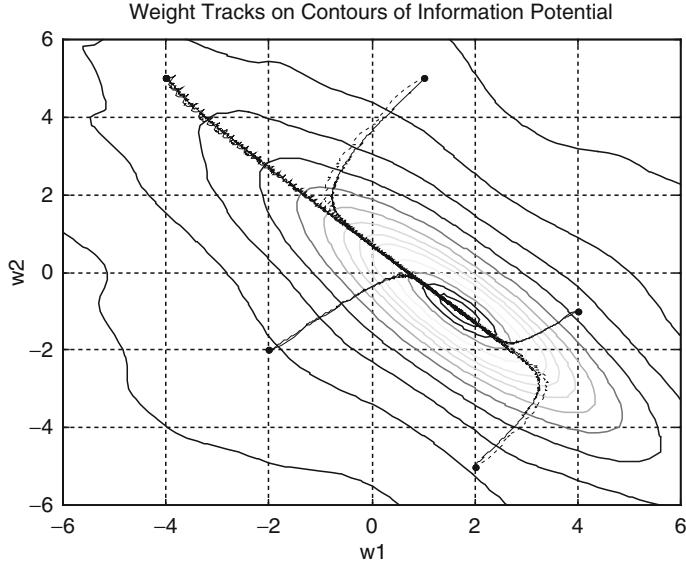
**Fig. 4.3.** Effects of the window length in adaptation.

the data, convergence actually does not happen because there is insufficient information to estimate the gradient of the IP ( $\sigma = 0.1$ ). Figure 4.3 shows the dependence of SIG on the window length.

For this problem where the solution is obtained with zero error, we see that smaller windows provide faster convergence. However, if the error was not zero, misadjustment would increase with smaller  $L$  (not shown).

In the following simulation we illustrate the performance of SIG(1) in training a linear filter; once again, for visualization purposes, we have chosen a 2-weight adaptive filter. The goal is to predict a time series from its most recent two samples, where the sequence is generated by  $x(t) = \sin 20t + 2 \sin 40t + 3 \sin 60t$  sampled at 100 Hz. The training set consists of 32 samples, which approximately corresponds to one period of the signal, and it is repeated for 150 epochs for both SIG(1) and LMS algorithms.

The weight tracks of both algorithms starting from five different initial conditions are shown in Figure 4.4 along with the contours for the minimum error entropy criterion for this training set. Therefore, we can expect that in linear filtering when the signals are stationary and the initializations are close to the minimum, there is a  $\sigma$  that makes the MEE and the MSE converge at the same speed of adaptation and misadjustment. But in general the two algorithms perform differently because of the changes in performance surfaces.



**Fig. 4.4.** Weight tracks for SIG (solid) and LMS (dotted) in online training a linear filter for time-series prediction (from [88]).

### SIG(1) and Shannon's Entropy Gradient Estimation

This section shows the difficulty of analyzing the stochastic gradient algorithm. We have stated that working with the IP or Renyi's entropy is equivalent if the goal is to find the stationary points of the cost, but we can obviously derive formulas to minimize Renyi's entropy directly. It is just a matter of incorporating the logarithm in the cost function of Eq. (3.12) and computing the partial derivative. This equivalence is true for the estimators of IP with double sums, and if the stochastic gradient is a consistent estimator of IP it should also be true for the SIG. Unfortunately, this is not the case as we now demonstrate.

The SIG estimate of  $\alpha$ -Renyi's entropy from Eq. (4.12) yields immediately

$$\hat{H}_\alpha(X) = \frac{1}{1-\alpha} \log \left[ \frac{1}{L} \sum_{i=n-L}^{n-1} \kappa_\sigma(x_n - x_i) \right]^{\alpha-1} = -\log \left[ \frac{1}{L} \sum_{i=n-L}^{n-1} \kappa_\sigma(x_n - x_i) \right] \quad (4.16)$$

Notice that the entropy order disappears from the expression for the SIG class of estimators of  $\alpha$ -Renyi's entropy. This means that the SIG algorithms when used to estimate  $\alpha$ -Renyi's entropy are unable to weight dense and sparse regions of the PDFs differently, therefore they provide for  $\sigma \rightarrow 0$ ,  $L \rightarrow \infty$  the same result as  $\hat{H}_2(X)$ .

This result can also be interpreted as an unbiased estimator of the gradient of Shannon's entropy using Parzen windowing.

**Theorem 4.1.** *The expected value of the stochastic gradient of Renyi's  $\alpha$ -entropy in Eq. (4.16) is an unbiased instantaneous estimator of the gradient of Shannon's entropy estimated using Parzen windowing.*

*Proof.* Consider Shannon's entropy of the error given by  $H_S(e) = -E[\log p_e(e)]$ . Suppose we estimate this quantity by substituting the PDF with its Parzen window estimate over the most recent  $L$  samples at time  $k$ . Then this estimate of Shannon's entropy at time  $n$  is given by

$$\hat{H}_{S,n}(e) \approx -E \left[ \log \left( \frac{1}{L} \sum_{i=n-L}^{n-1} \kappa_\sigma(e_n - e_i) \right) \right]. \quad (4.17)$$

The error gradient of Eq. (4.17) is easily determined to be

$$\frac{\partial \hat{H}_{S,n}}{\partial (e_n - e_i)} = E \left[ \frac{\sum_{i=n-L}^{n-1} \kappa'_\sigma(e_n - e_i)}{\sum_{i=n-L}^{n-1} \kappa_\sigma(e_n - e_i)} \right], \quad (4.18)$$

which is the expected value of the SIG estimator of  $\alpha$ -Renyi's entropy of Eq. (4.16).

There is even a stranger special case of the stochastic gradient that occurs when the window length  $L = 1$ , the kernel function is a Gaussian and the goal is to adapt a linear combiner. For Gaussian kernels, the derivative of the kernel can be written in terms of the kernel function itself as  $G'_\sigma(x) = -xG_\sigma(x)/\sigma^2$ . When these are substituted in Eq. (4.18), and the chain rule is used to compute the gradient with respect to the weights of the linear combiner, the solution becomes

$$\frac{\partial \hat{H}_{S,n}}{\partial w_k} = -\frac{1}{\sigma^2}(e(n) - e(n-1))(x_k(n) - x_k(n-1)). \quad (4.19)$$

Notice two things: First, with respect to the SIG(1) applied to the IP in Eq. (4.15), the nonlinear weighting produced by the Gaussian kernel disappears. Second, notice the resemblance between this update and the LMS update, which is given by  $-2e(n)x_k(n)$ , for the linear combiner. The updates in Eq. (4.19) are based on the instantaneous increments of the signal values in this special case, whereas the LMS updates are based on the instantaneous signal values. Therefore, it is unclear if indeed Eq. (4.19) is a proper stochastic gradient estimator of Shannon's entropy because the weighting produced by the kernels actually disappears from the update (see Eq. (4.15)). Further work is necessary.

## 4.4 Self-Adjusting Stepsize for MEE (MEE-SAS)

From the analysis of the shape of the error entropy cost function in Section 3.5 we foresee changes in curvature (i.e., varying Hessian) even for the scalar case. Constant stepsize gradient algorithms simply cannot take advantage of such characteristics, so an effort to develop variable stepsize algorithms is in order, or even second order search algorithms. Here we deal with the variable stepsize case [128]. As can be easily inferred from the definition of the information potential Eq. (3.12), the largest value of the potential occurs when all the samples are at the same location ( $V(0)$ ): that is,  $V(e) \leq V(0)$  and  $V(0)$  provides an upper bound on the achievable  $V(e)$ . Seen from a different perspective,  $V(0)$  is the ideal “target” value for the information potential curve during adaptation, which will be reached when there is no observation noise and the system has sufficient degrees of freedom to achieve an all-zero error. Thus  $[\hat{V}(0) - \hat{V}(e)]$  is always a nonnegative scalar quantity that does not change the direction of the gradient and can be used to accelerate the conventional gradient search algorithm given in Eq. (3.16). Note that this quantity is the numerator of the normalized IP of Eq. (3.56). This modified search algorithm is named MEE-self-adjusting stepsize (SAS) because it actually works as a selfadjusting stepsize algorithm as we show below. Let us define the new cost function as

$$\min_{\mathbf{w}} J_{\text{MEE-SAS}}(e) = [\hat{V}(0) - \hat{V}(e)]^2. \quad (4.20)$$

Taking the gradient of Eq. (4.20) directly yields

$$\nabla J_{\text{MEE-SAS}}(e) = -2[\hat{V}(0) - \hat{V}(e)] \cdot \nabla \hat{V}(e),$$

and the weight update for MEE-SAS becomes

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu[\hat{V}(0) - \hat{V}(e)]\nabla \hat{V}(e) = \mathbf{w}(n) + \mu(n)\nabla \hat{\mathbf{V}}(e), \quad (4.21)$$

where  $\mu(n) = \mu[\hat{V}(0) - \hat{V}(e)]$ . It is also easy to show that the MEE-SAS cost function preserves the location of the MEE minimum. In fact, stationary points are not affected by monotonic transformations on the cost. In MEE-SAS,  $f(\hat{V}(e)) = [\hat{V}(0) - \hat{V}(e)]^2$  is monotonic, therefore the optimum coincides with the MEE solution. Notice that Eq. (4.21) basically states that we have a gradient descent procedure with an adaptive stepsize, which is automatically determined by how different the actual IP is from the maximum. We can expect that this new cost function and corresponding gradient descent will automatically take care of the differences in curvature encountered when searching the IP cost function that were mentioned in the analysis of Section 3.5: giving acceleration when far away from the optimal solution and automatically reducing the stepsize as the solution is approached. Using the same quadratic approximation of Section 3.5 for the performance surface, it

is easy to show that to ensure convergence of the MEE-SAS algorithm, a necessary condition is

$$0 < \mu(n) < \frac{-2}{\lambda_k}, \quad (4.22)$$

where  $\mu(n) = \mu[V(0) - V(e)]$  and  $\lambda_k$  is the largest eigenvalue of the MEE cost function. The intuition about the role of  $\mu(n)$  can be mathematically proved as follows.

**Theorem 4.2.** *Let  $\tilde{\Xi}$  and  $\Xi$  denote the Hessian approximation of MEE-SAS and MEE, respectively. The relation between these Hessians is the following,*

$$\tilde{\Xi} = -c\Xi + (\tilde{\mathbf{w}}(n)^T \Xi \tilde{\mathbf{w}}(n))\Xi + 2\Xi \tilde{\mathbf{w}}(n) \tilde{\mathbf{w}}(n)^T \Xi^T, \quad (4.23)$$

where  $c = 2[\hat{V}(0) - \hat{V}_{\mathbf{w}_*}(e)]$ .

*Proof.* Differentiating Eq. (4.20) twice with respect to the weight vector produces  $\tilde{\Xi} = -2[\hat{V}(0) - \hat{V}(e)]\nabla^2\hat{V}(e) + 2\nabla\hat{V}(e)\nabla\hat{V}(e)^T$  and substituting the second-order Taylor series approximation of IP around the optimal solution  $\hat{V}(e) = \hat{V}_{\mathbf{w}_*}(e) + 1/2\tilde{\mathbf{w}}(n)^T \Xi \tilde{\mathbf{w}}(n)$  and  $\nabla\hat{V}(e) = -\Xi \tilde{\mathbf{w}}(n)$  yields the expected result for Eq. (4.23).

From this theorem and using the eigendecompositions of MEE-SAS ( $\tilde{\Xi} = \tilde{\mathbf{Q}}\tilde{\Lambda}\tilde{\mathbf{Q}}^T$ ) and MEE ( $\Xi = \mathbf{Q}\Lambda\mathbf{Q}^T$ ) with a coordinate transformation into the natural modes, we obtain

$$\begin{aligned} \tilde{\mathbf{Q}}\tilde{\Lambda}\tilde{\mathbf{Q}}^T &= -c\mathbf{Q}\Lambda\mathbf{Q}^T + (\tilde{\mathbf{w}}(n)^T \mathbf{Q}\Lambda\mathbf{Q}^T \tilde{\mathbf{w}}(n))\mathbf{Q}\Lambda\mathbf{Q}^T \\ &\quad + 2\mathbf{Q}\Lambda\mathbf{Q}^T \tilde{\mathbf{w}}(n) \tilde{\mathbf{w}}(n)^T (\mathbf{Q}\Lambda\mathbf{Q}^T)^T \\ &= \mathbf{Q}[-c\Lambda + (\tilde{\mathbf{w}}(n)^T \mathbf{Q}\Lambda\mathbf{Q}^T \tilde{\mathbf{w}}(n))\Lambda + 2\Lambda\mathbf{Q}^T \tilde{\mathbf{w}}(n) \tilde{\mathbf{w}}(n)^T Q\Lambda^T]\mathbf{Q}^T \\ &= \mathbf{Q}[-c\Lambda + (\mathbf{v}(n)^T \Lambda \mathbf{v}(n))\Lambda + 2\Lambda \mathbf{v}(n) \mathbf{v}(n)^T \Lambda]\mathbf{Q}^T, \end{aligned} \quad (4.24)$$

where  $\mathbf{v}(n) = \mathbf{Q}^T \tilde{\mathbf{w}}(n)$ . In Eq. (4.24) substituting the eigendecomposition of the matrix in brackets denoted by  $\Sigma\mathbf{D}\Sigma$ , where  $\Sigma$  is orthonormal and  $\mathbf{D}$  is diagonal, yields

$$\tilde{\mathbf{Q}}\tilde{\Lambda}\tilde{\mathbf{Q}}^T = \mathbf{Q}\Sigma\mathbf{D}\Sigma^T\mathbf{Q}^T. \quad (4.25)$$

By direct comparison, its eigenvectors and the eigenvalues are determined to be

$$\tilde{\mathbf{Q}} = \mathbf{Q}\Sigma, \quad \tilde{\Lambda} = \mathbf{D}. \quad (4.26)$$

The entries in  $\Sigma\mathbf{D}\Sigma^T$  are found as follows. The  $i$ th diagonal entry is  $-c\lambda_i + (\sum_{j=1}^M \lambda_j v_j^2) \lambda_i + 2\lambda_i^2 v_i^2$  and the  $(i, j)$ th entry is  $2\lambda_i \lambda_j v_i v_j$ , where  $\lambda_i$  is the  $i$ th diagonal entry of  $\Lambda$  and  $v_i$  the  $i$ th entry of  $\mathbf{v}(n)$ .

Especially if  $\mathbf{v}(n)$  is small, the matrix  $[-c\Lambda + (\mathbf{v}(n)^T \Lambda \mathbf{v}(n))\Lambda + 2\Lambda \mathbf{v}(n) \mathbf{v}(n)^T \Lambda]$  is diagonally dominant; hence (due to the Gershgorin theorem [192]) its eigenvalues will be close to those of  $-c\Lambda$ . In addition, the eigenvectors will also be close to identity.

Consider the special case when the operating point is moving along the  $k$ th eigenvector ( $\mathbf{v} = [0, \dots, v_k, \dots, 0]^T$ ). Then the expressions simplify to:

$$\tilde{\Lambda} = \begin{bmatrix} -c\lambda_1 + \lambda_k\lambda_1v_k^2 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & -c\lambda_2 + \lambda_k\lambda_2v_k^2 & & 0 & & 0 \\ \vdots & & \ddots & & & \vdots \\ 0 & 0 & & -c\lambda_k + 3\lambda_k^2v_k^2 & & 0 \\ \vdots & & & & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & -c\lambda_M + \lambda_k\lambda_Mv_k^2 \end{bmatrix} \quad (4.27)$$

In real scenarios, there exist modes that converge more slowly than others due to the eigenvalue spread. If we analyze the convergence along the principal axis of  $\Xi$ , we obtain

$$\begin{aligned} \tilde{\lambda}_j &= -2[\hat{V}(0) - \hat{V}_{\mathbf{w}_*}(e)]\lambda_j + \lambda_k\lambda_jv_k^2, \quad \forall j \neq k \\ \tilde{\lambda}_k &= -2[\hat{V}(0) - \hat{V}_{\mathbf{w}_*}(e)]\lambda_k + \lambda_k^2v_k^2 + 2\lambda_k^2v_k^2 \\ &= -2[\hat{V}(0) - \hat{V}_{\mathbf{w}_*}(e)]\lambda_k + 3\lambda_k^2v_k^2, \quad j = k \end{aligned} \quad (4.28)$$

When the weights are close to the optimal solution  $v_k^2 \approx 0$ , the eigenvalues of MEE-SAS are proportional to the eigenvalues of the MEE cost which is quadratic. On the other hand, when the weights of MEE-SAS are far from the solution,  $v_k^2$  is large and thus the second term in Eq. (4.28) dominates and the eigenvalues are proportional to the square of the original eigenvalues. A consequence of this is that MEE-SAS cost function has the remarkable property of changing curvature depending upon the distance from the optimal solution. Also note that the convergence along the  $k$ th natural mode is faster than other modes due to the extra  $2\lambda_k^2v_k^2$  term when the weights are far from the optimal solution.

For each natural mode  $v_k$  in Eq. (4.28), the relationship between the eigenvalue of MEE-SAS ( $\tilde{\lambda}_k$ ) and that of MEE ( $\lambda_k$ ) is

$$\tilde{\lambda}_k = -2[\hat{V}(0) - \hat{V}_{\mathbf{w}_*}(e)]\lambda_k + 3\lambda_k^2v_k^2 = -\lambda_k(c - 3\lambda_kv_k^2), \quad (4.29)$$

where  $c = 2[\hat{V}(0) - \hat{V}_{\mathbf{w}_*}(e)]$  is nonnegative. Because we maximize the cost function  $V(\mathbf{e})$  in MEE, the eigenvalues  $\lambda_k$  of its Hessian approximation are negative. Similarly, for MEE-SAS, the minimization of its cost function makes  $\tilde{\lambda}_k$  positive. The turning point of curvature occurs when  $\tilde{\lambda}_k = -\lambda_k$ , or

$$c - 3\lambda_kv_k^2 = 1. \quad (4.30)$$

From Eq. (4.30), we specifically obtain

$$v_k = \pm\sqrt{\frac{c-1}{3\lambda_k}}. \quad (4.31)$$

Using the nonnegative property of  $c$  in Eq. (4.31), we conclude that  $0 \leq c \leq 1$  for real eigenvalues, therefore,  $c = 0 \rightarrow V(0) = V_{\mathbf{w}_*}(e)$ , whereas  $c = 1 \rightarrow v_k = 0$  (i.e.,  $\mathbf{w} = \mathbf{w}_*$ ).

It is interesting to note that the point at which the curvature changes from higher than second order to second order is closer to the optimal solution when  $V(\mathbf{0}) \neq V_{\mathbf{w}_*}(\mathbf{e})$  than when  $V(0) = V_{\mathbf{w}_*}(e)$ . In fact, when  $V(0) = V_{\mathbf{w}_*}(e)$ , the turning point of curvature is  $v_k = \pm\sqrt{-1/3\lambda_k}$ , whereas it is  $v_k = \pm\sqrt{(c-1)/3\lambda_k}$ , when  $V(0) \neq V_{\mathbf{w}_*}(e)$ .

Thus, the curvature turning point  $v_k$  is farther away from the optimal solution when the achievable error is zero than for cases where the final error is nonzero. Because  $v_k$  marks the change of curvature from fourth order to second order, this implies that for practical scenarios (i.e.,  $V(0) \neq V_{\mathbf{w}_*}(e)$ ), the curvature is going to be predominately fourth order, leading to much faster convergence than MEE for the same initial step size.

### Switching Scheme between MEE-SAS and MEE

One disadvantage of MEE-SAS over the MEE with the same stepsize for tracking is the smaller gradient of the performance surface near the optimum which makes it slow to track changes in the input statistics (although both algorithms have quadratic performance surfaces near the optimum, the eigenvalues are different; see Eq. (4.28)). This was observed in practical problems (prediction of nonstationary Mackey-Glass (MG) time series) that require a continuous fine adjustment of the optimal weight vector. Combining MEE and MEE-SAS algorithms for nonstationary signals becomes therefore important. In order to decide the switching time to maximize convergence speed, an analytical criterion needs to be developed.

The dynamics of adaptation can be understood in terms of energy minimization in the context of Lyapunov stability theory [206]. This method confirms the intuition that the best strategy for switching is when the gradients of the two adaptation schemes are identical:

$$|\nabla \mathbf{J}_{\text{MEE-SAS}}| = |\nabla \mathbf{J}_{\text{MEE}}|. \quad (4.32)$$

Therefore, in the region satisfying the condition  $|\nabla \mathbf{J}_{\text{MEE-SAS}}| > |\nabla \mathbf{J}_{\text{MEE}}|$ , MEE-SAS should be used because MEE-SAS converges faster than MEE, otherwise MEE is used. However, the application of this switching decision implies a high computational complexity inasmuch as both algorithms need to be run in parallel. Instead, we can evaluate Eq. (4.32) to obtain the condition for which  $|\nabla J_{\text{MEE-SAS}}| > |\nabla J_{\text{MEE}}|$  as

$$\begin{aligned} 4\mu_{\text{MEE-SAS}} [V(0) - V(e)]^2 \left\| \frac{\partial V(e)}{\partial \mathbf{w}} \right\|^2 &> \mu_{\text{MEE}} \left\| \frac{\partial V(e)}{\partial \mathbf{w}} \right\|^2 \\ \Leftrightarrow V(e) < V(0) - \frac{1}{2} \sqrt{\frac{\mu_{\text{MEE}}}{\mu_{\text{MEE-SAS}}}}. \end{aligned} \quad (4.33)$$

Therefore one needs just to evaluate the information potential at each iteration and compare it with a constant, which is a function of the learning rates of MEE and MEE-SAS. This provides a very practical test and only requires running a single algorithm. But of course, the selection of the stepsize for each method still requires experimentation.

## Curvature Analysis of MEE and MEE-SAS

### Effect of Kernel Size in Perfect Identification

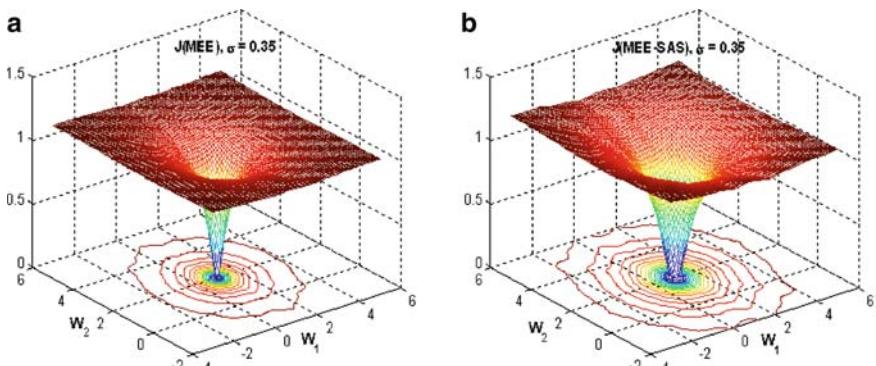
The default system identification example is used here. This case study aims to illustrate how the MEE ( $\hat{V}(0) - \hat{V}_{\mathbf{w}_*}(e)$ ) and MEE-SAS ( $[\hat{V}(0) - \hat{V}_{\mathbf{w}_*}(e)]^2$ ) cost functions are altered as a consequence of changing the kernel size in the estimator. Figure 4.5 shows both performance surfaces in 3D. One hundred noiseless training samples are utilized to obtain the contour and gradient vector plots. The kernel size is set to  $\sigma = 0.35$ .

Our interest is in illustrating the gradient differences between the two cost functions, therefore we plot the gradient difference between MEE from MEE-SAS in Figure 4.6.

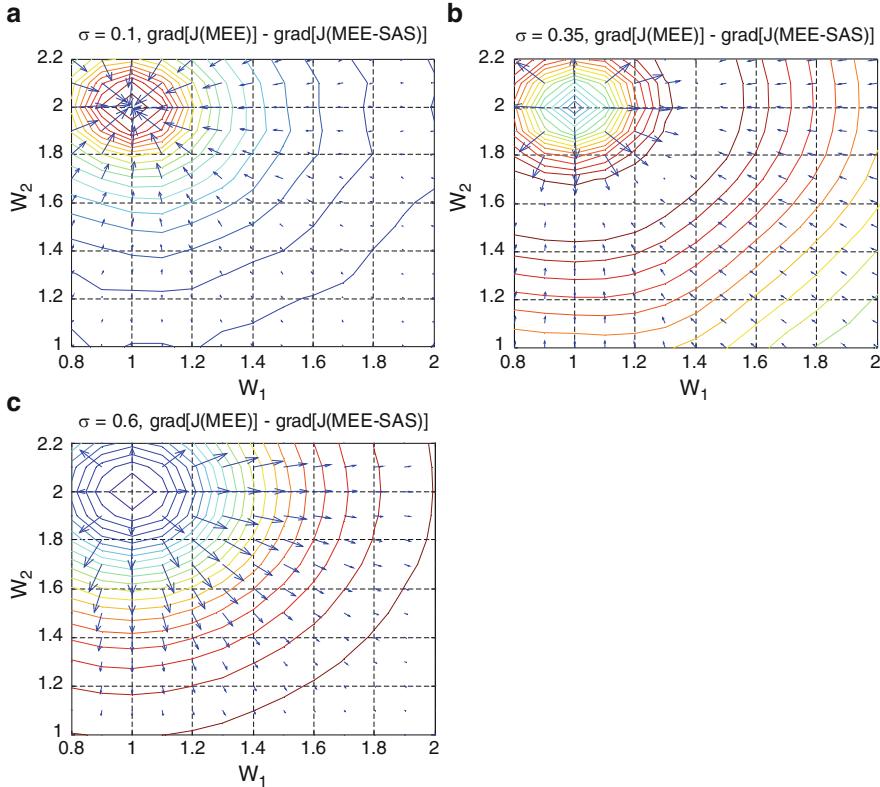
As illustrated in Figure 4.6, when using a small kernel size ( $\sigma = 0.1$ ), MEE-SAS has a larger gradient magnitude when compared with MEE over most of the space whereas for a larger kernel size ( $\sigma = 0.6$ ), the position reverses (MEE gradient is larger).

### Effect of Kernel Size in Imperfect Identification

We just show the simulation result for the measurement noise case (similar results are obtained when the modeling error is nonzero). We add uniform distributed noise with three different powers ( $P = 1, 2$ , and  $3$ ) to the desired signal of the above example.



**Fig. 4.5.** Performance surface of normalized MEE (a) and MEE-SAS (b) (from [128]).

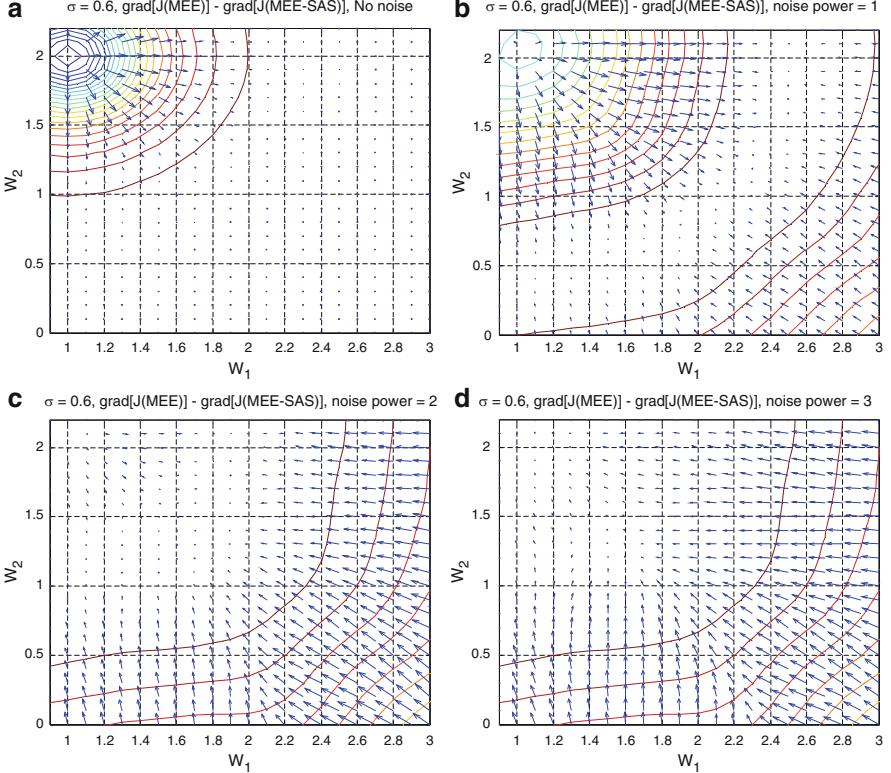


**Fig. 4.6.** Contour and gradient difference between MEE and MEE-SAS on error information potential for various choices of kernel size (from [128]). : (a) 0.1; (b) = 0.35; (c) 0.6.

As observed in Figure 4.7, the higher the noise power, the larger is the region over which the MEE-SAS gradient exceeds the MEE, as our analysis showed. This means that when  $\hat{V}(0) \neq \hat{V}_{\mathbf{w}_*}(e)$  the point at which the curvature changes from higher than second order to second order is closer to the optimal solution than in perfect identification. This also means that the larger the final error, the faster is the convergence.

## 4.5 Normalized MEE (NMEE)

One of the difficulties of the gradient-based algorithms is that an increase in input power appears directly in the weight update formulas, and can momentarily cause divergence (i.e. very large change in the weights). Divergence in an adaptive algorithm is catastrophic because all the information about past data is contained in the weights of the filter. If the weights diverge, then it is



**Fig. 4.7.** Contour and gradient difference between MEE and MEE-SAS on error information potential for three different measurement noises (from [128]). : (a) no noise; (b) noise power 1; (c) noise power 2; (d) noise power 3.

equivalent to throwing away all of the past data and starting again! This is why ways to avoid divergence have been sought in adaptive filtering. One of the most widely used methods is intuitive and consists in dividing the step-size by the norm of the current input vector (values in the filter taps) [253]. This procedure, called the normalized LMS algorithm, can also be proved mathematically as the best strategy to have an update that is insensitive to fluctuations of input power [143].

We derive the NMEE algorithm by analogy with the normalized LMS (NLMS) algorithm, that is, as a modification of the ordinary MEE criterion in light of the principle of minimal disturbance [143]. The criterion of NMEE is formulated as the following constrained optimization,

$$\min \|\mathbf{w}(n+1) - \mathbf{w}(n)\|^2 \quad \text{subject to} \quad \hat{V}(0) - \hat{V}(e_p(n)) = 0, \quad (4.34)$$

where  $e_p(n) = z(n) - \mathbf{w}(n+1)^T x(n)$  is the posterior error, and Eq. (4.34) translates the constraint of optimal performance in terms of the information potential.

The above constrained optimization problem can be solved using the method of Lagrange multipliers as follows,

$$J(e(n)) = \|\mathbf{w}(n+1) - \mathbf{w}(n)\|^2 + \lambda(\hat{V}(0) - \hat{V}(e_p(n))). \quad (4.35)$$

Taking the derivative with respect to  $\mathbf{w}(n+1)$  and equating the result to zero to find the minimum we obtain

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{1}{2}\lambda \nabla \hat{\mathbf{V}}(e_p(n)). \quad (4.36)$$

To solve for the unknown  $\lambda$ , Eq. (4.36) is substituted into the constraint of Eq. (4.34) i.e.,  $\hat{V}(0) = \hat{V}(e_p(n)) \rightarrow e_p(n) = e_p(i)$ ,  $n-L < i < n-1$ , to yield

$$\lambda = \frac{2 \sum_{i=n-L}^{n-1} (e_a(n) - e_a(i))}{\nabla \hat{\mathbf{V}}(e_p(n))^T \left[ \sum_{i=n-L}^{n-1} (\mathbf{x}(n) - \mathbf{x}(i)) \right]} \quad (4.37)$$

which when plugged back into Eq. (4.36) yields

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta \frac{\left[ \sum_{i=n-L}^{n-1} (e_a(n) - e_a(i)) \right] \nabla \hat{\mathbf{V}}(e_p(n))}{\nabla \hat{\mathbf{V}}(e_p(n))^T \left[ \sum_{i=n-L}^{n-1} (\mathbf{x}(n) - \mathbf{x}(i)) \right]}, \quad (4.38)$$

where  $e_a(n) = z(n) - \mathbf{w}(n)^T \mathbf{x}(n)$  is the a priori error, and  $\eta$  is the stepsize that can be proven to be between 0 and 2 for stability. In this update, there is an added difficulty because estimating  $\mathbf{w}(n+1)$  requires the a posteriori error  $e_p(n)$ . We approximate it by  $e_a(n)$  because the algorithm minimizes  $\|\mathbf{w}(n+1) - \mathbf{w}(n)\|^2$ , so the approximation gets progressively better through adaptation. Therefore, the weight update for NMEE is

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta \frac{\left[ \sum_{i=n-L}^{n-1} (e_a(n) - e_a(i)) \right] \nabla \hat{\mathbf{V}}(e_a(n))}{\nabla \hat{\mathbf{V}}(e_a(n))^T \left[ \sum_{i=n-L}^{n-1} (\mathbf{x}(n) - \mathbf{x}(i)) \right]}. \quad (4.39)$$

The Lagrange multiplier in LMS is just the error divided by the input power estimated with the samples in the taps. Here Eq. (4.37) shows that the situation for NMEE is more complex, because not only does the input power appear in the denominator but also the information forces (the gradient of the error). Due to this division by the information forces and input power, we

can expect that the sensitivity to the input power and the kernel size will be diminished in NMEE.

When the SIG(1) is utilized in Eq. (4.39) the stochastic update for the NMEE algorithm becomes

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) + \eta \frac{(e_a(n) - e_a(n-1)) \nabla \hat{\mathbf{V}}(e_a(n))}{\left(\nabla \hat{\mathbf{V}}(e_a(n))\right)^T (\mathbf{x}(n) - \mathbf{x}(n-1))} \\ &= \mathbf{w}(n) + \eta \frac{(e_a(n) - e_a(n-1)) (\mathbf{x}(n) - \mathbf{x}(n-1))}{\|\mathbf{x}(n) - \mathbf{x}(n-1)\|^2}.\end{aligned}\quad (4.40)$$

We see again (see Section 4.3) that the kernel disappears in this expression, so there is no local weighting on the errors. Moreover, this is exactly the normalized LMS update with the instantaneous input and error substituted by the product of their increments, which was previously derived as the error-whitening LMS [258].

To interpret the results for  $L > 1$  we write the equations in component form that generalize for a higher number of samples. For  $L = 2$  and a one-tap filter the update becomes

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta \frac{(\Delta e(2) + \Delta e(1)) \begin{bmatrix} \Delta x_1(2) \\ \Delta x_2(2) \end{bmatrix} \begin{bmatrix} \Delta x_1(1) \\ \Delta x_2(1) \end{bmatrix}^T \begin{bmatrix} G(2) \Delta e(2) \\ G(1) \Delta e(1) \end{bmatrix}}{[G(2) \Delta e(2) \ G(1) \Delta e(1)] \left\| \begin{bmatrix} \Delta x_1(2) \\ \Delta x_2(2) \end{bmatrix} \begin{bmatrix} \Delta x_1(1) \\ \Delta x_2(1) \end{bmatrix} \right\|^2 \begin{bmatrix} 1 \\ -1 \end{bmatrix}}, \quad (4.41)$$

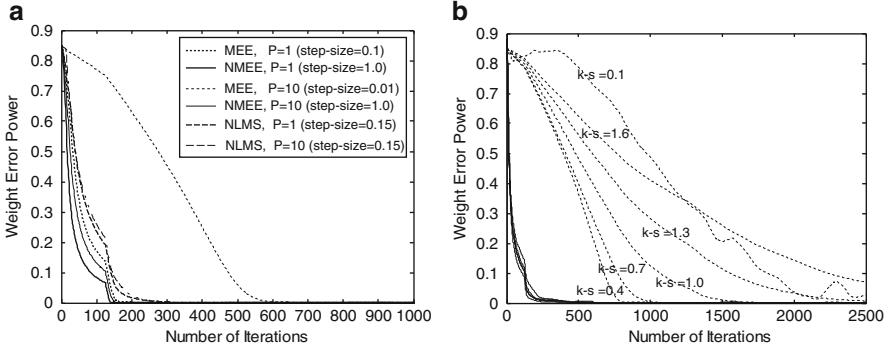
where we are using the simplified index notation  $\Delta e(i) = e(n) - e(n-i)$ . We can observe that there is an error term in the numerator that is not affected by the kernel, but the kernel does not cancel out completely, which suggests that the effect of the kernel size exists but will be decreased with respect to the MEE.

### NMEE Dependence on Input Power

We consider a moving-average plant with transfer function given by (order = 9)

$$\begin{aligned}H(z) &= 0.1 + 0.2z^{-1} + 0.3z^{-2} + 0.4z^{-3} + 0.5z^{-4} + 0.4z^{-5} + 0.3z^{-6} \\ &\quad + 0.2z^{-7} + 0.1z^{-8}.\end{aligned}$$

The FIR adaptive filter is selected with equal order to the plant. A standard method of comparing the performance in system identification is to plot the weight error norm inasmuch as this is directly related to misadjustment [143]. In each case the power of the weight noise (averaged over 125 samples) was plotted versus the number of iterations. The input to both the plant and the adaptive filter is white Gaussian noise. In the first experiment, a unit power



**Fig. 4.8.** (a) Average weight error power for MA(9) comparing the MEE (dashed lines), NMEE (solid line), and NLMS for different input powers. (b) Average weight error power for MA(9) with different kernel sizes, with white Gaussian input of variance 10, eigenspread  $S = 550$ . (kernel sizes left to right dashed lines: 0.4, 0.7, 1, 1.3, 1.6, 0.1) (from [130]).

input is selected (1P), whereas in the second experiment the input is  $10 \times$  power (10P). The kernel size using Silverman's rule is  $\sigma = 0.7$ .

In this perfect identification scenario, the filter exactly tracks the output of the plant. Figure 4.8a shows the plot of the weight error norm for a moving average model. We choose a large stepsize within the range of MEE stability. For the unit input power case, both MEE and NMEE converge in 190 iterations with basically the same misadjustment. To guarantee the stability of MEE adaptation for  $10 \times$  input power, the stepsize is chosen 10 times smaller, and it remains at the same value for NMEE. NMEE just takes 190 iterations to converge as compared to MEE which takes nearly 700 iterations with practically the same misadjustment of  $1.243 \times 10^{-8}$ . We can therefore conclude that the NMEE is insensitive to the input power, as expected. We also present in Figure 4.8a the convergence of the NLMS for the same misadjustment, and conclude that the NMEE is faster than the NLMS.

### NMEE Dependence on Kernel Size and Speed

Selecting a proper kernel size is one important factor because it depends on the error statistics and it influences the performance of the MEE algorithm. The effect of kernel sizes on both MEE and NMEE is shown in Figure 4.8b, where the weight error power curves with different kernel sizes are plotted when the input data is white Gaussian with zero mean,  $10 \times$  unit variance and with an eigenvalue spread ratio of  $S = 550$ . We observe that the performance of MEE is sensitive to the kernel size and this sensitivity increases when the eigenvalue spread of the input signal increases (not shown). However, NMEE shows a much more uniform performance with different kernel sizes even when the input has a large dynamic range. The misadjustment of NMEE in the worst case is almost the same as that of MEE in the best case. Furthermore,

the kernel size of 0.7 and 1 for MEE and NMEE, respectively, are found to be optimal, giving the lowest misadjustment of  $6.7 \times 10^{-10}$  and  $8.2 \times 10^{-14}$  in the case when  $S = 1$  and  $3.8 \times 10^{-8}$  and  $1.5 \times 10^{-12}$  when  $S = 550$ .

Another important aspect of these experiments is the different speed of convergence between the MEE and the NMEE. Figure 4.8b clearly shows that there are two sets of curves, one for each algorithm. But notice that the misadjustment between the two algorithms is practically the same ( $8.2 \times 10^{-14}$  versus  $6.7 \times 10^{-10}$ ), therefore we conclude that the NMEE is substantially faster than the MEE. However, the robustness to outliers of NMEE may be compromised.

## 4.6 Fixed-Point MEE (MEE-FP)

Until now only first-order (gradient-based) algorithms for EEC were developed. In this section we present a fixed-point algorithm to solve the EEC criterion as a continuation of the discussion in Section 3.5. As is well known, fixed-point algorithms have second-order convergence to the optimal solution, as the well-known recursive least square (RLS) algorithm of adaptive filtering [143], and they do not have free parameters. However, the region of convergence for the fixed-point updates must be determined and fast solutions are not always available. To complement the analysis in that section we present the algorithms for the Gaussian kernel in vector notation.

Given the criterion  $V(e)$  of Eq. (4.16), and without assuming any prior knowledge about the location of its minimizing argument  $\mathbf{w}_*$ , we wish to devise a procedure that starts from an initial guess for  $\mathbf{w}_*$  and then improves upon it in a recursive manner until ultimately converging to  $\mathbf{w}_*$ . The stationary point  $\mathbf{w}$  obeys

$$\frac{\partial V(e)}{\partial \mathbf{w}_k} \Big|_{w_k=w^*} = \frac{1}{2N^2\sigma^2} \sum_{i=1}^N \sum_{j=1}^N (e_i - e_j) G_{\sigma\sqrt{2}}(e_i - e_j)(\mathbf{x}_i - \mathbf{x}_j) = 0. \quad (4.42)$$

The condition in Eq. (4.42) implies that at  $\mathbf{w}_k = \mathbf{w}_*$  an optimum exists i.e.,  $e_1 = \dots = e_N$ . Substituting  $e_i - e_j = z_i - z_j - (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{w}$  in Eq. (4.42) yields directly

$$\begin{aligned} \mathbf{w}_* &= \left( \sum_{i=1}^N \sum_{j=1}^N G_{\sigma\sqrt{2}}(e_j - e_i)[\mathbf{x}_j - \mathbf{x}_i][\mathbf{x}_j - \mathbf{x}_i]^T \right)^{-1} \\ &\quad \left( \sum_{i=1}^N \sum_{j=1}^N G_{\sigma\sqrt{2}}(e_j - e_i)[z_j - z_i][\mathbf{x}_j - \mathbf{x}_i] \right). \end{aligned} \quad (4.43)$$

Equation (4.43) looks very similar to the least square optimal solution (for comparison we should substitute the input and desired response vectors by their time increments, and apply a weighting to every term given by the

Gaussian of the incremental errors). But in fact it is rather different, because the error in Eq. (4.43) is still a function of the weights, and must be computed iteratively from the current weights. Therefore Eq. (4.43) is in fact a recursive algorithm normally called a fixed-point update. Let the weight vector  $\mathbf{w}_k$  at iteration  $k$  be the estimate of the optimal solution. Then, the estimate of the new weight vector at iteration  $k + 1$  becomes

$$\begin{aligned}\mathbf{w}_{k+1} = F(\mathbf{w}_k) &= \left( \frac{1}{2k^2\sigma^2} \sum_{i=1}^k \sum_{j=1}^k G_{\sigma\sqrt{2}}(e_j - e_i)[\mathbf{x}_j - \mathbf{x}_i][\mathbf{x}_j - \mathbf{x}_i]^T \right)^{-1} \\ &\quad \left( \frac{1}{2k^2\sigma^2} \sum_{i=1}^k \sum_{j=1}^k G_{\sigma\sqrt{2}}(e_j - e_i)[z_j - z_i][\mathbf{x}_j - \mathbf{x}_i] \right) \\ &= \mathbf{R}_\Delta(\mathbf{w})^{-1} \mathbf{P}_\Delta(\mathbf{w}),\end{aligned}\quad (4.44)$$

where  $\mathbf{R}_\Delta(\mathbf{w}_n)$  and  $\mathbf{P}_\Delta(\mathbf{w}_n)$  are the pairwise locally weighted incremental autocorrelation matrix and cross-correlation vector of the input and desired signals given by

$$\mathbf{R}_\Delta(\mathbf{w}_k) = \frac{1}{2k^2\sigma^2} \sum_{i=1}^k \sum_{j=1}^k G_{\sigma\sqrt{2}}(e_j - e_i)[\mathbf{x}_j - \mathbf{x}_i][\mathbf{x}_j - \mathbf{x}_i]^T \quad (4.45)$$

$$\mathbf{P}_\Delta(\mathbf{w}_k) = \frac{1}{2k^2\sigma^2} \sum_{i=1}^k \sum_{j=1}^k G_{\sigma\sqrt{2}}(e_j - e_i)[z_j - z_i][\mathbf{x}_j - \mathbf{x}_i]. \quad (4.46)$$

This algorithm is called the fixed-point minimum error entropy (MEE-FP) algorithm, which is analogous to the RLS update rule that tracks the Wiener solution at every update.

It is important when applying a fixed-point solution to establish the region of convergence for the algorithm, using, for instance, the well-known Banach–Caccioppoli contraction theorem [118]. We were able to prove the local convergence of MEE-FP using a Taylor series expansion truncated at the linear term for the gradient around the optimal weight vector [129]. For solutions far away from the optimal solution, there is as yet no theoretical proof of convergence. However, experimental results suggest that the algorithm can be used in general conditions.

### Recursive MEE-FP with Forgetting Factor

We now derive an online forgetting recursive estimator for the fixed-point MEE. Investigating the structure of the estimator for the entropy weighted incremental autocorrelation  $\mathbf{R}_\Delta(\mathbf{w}_n)$  and the entropy weighted incremental cross-correlation  $\mathbf{P}_\Delta(\mathbf{w}_n)$  of the entropy in Eqs. (4.45) and (4.46), we obtain a recursive formula to update their estimates when a new sample is acquired.

When a new sample  $n$  arrives,  $\mathbf{R}_\Delta(\mathbf{w}_{n-1})$  and  $\mathbf{P}_\Delta(\mathbf{w}_{n-1})$  are modified with the new input-output sample pair  $(\mathbf{x}_n, z_n)$  as

$$\mathbf{R}_\Delta(\mathbf{w}_n) = \frac{(n-1)^2}{2n^2\sigma^2} \mathbf{R}_\Delta(\mathbf{w}_{n-1}) + \frac{1}{n^2\sigma^2} \sum_{i=1}^{n-1} G_{\sigma\sqrt{2}}(e_n - e_i)[\mathbf{x}_n - \mathbf{x}_i][\mathbf{x}_n - \mathbf{x}_i]^T \quad (4.47)$$

$$\mathbf{P}_\Delta(\mathbf{w}_n) = \frac{(n-1)^2}{2n^2\sigma^2} \mathbf{P}_\Delta(\mathbf{w}_{n-1}) + \frac{1}{n^2\sigma^2} \sum_{i=1}^{n-1} G_{\sigma\sqrt{2}}(e_n - e_i)[z_n - z_i][\mathbf{x}_n - \mathbf{x}_i]. \quad (4.48)$$

This exact recursion is useful for estimating the recursively weighted incremental autocorrelation and cross-correlation of stationary signals, however, it is time consuming (it increases with  $n$ ) and not suitable for nonstationary environments due to its increasing memory depth. Therefore, a forgetting recursive estimator using the past  $L$  samples is necessary to overcome both shortcomings. We define

$$\bar{\mathbf{R}}_\Delta(\mathbf{w}_n) = (1 - \lambda)\bar{\mathbf{R}}_\Delta(\mathbf{w}_{n-1}) + \frac{\lambda}{L} \sum_{i=n-L}^{n-1} G_{\sigma\sqrt{2}}(e_n - e_i)[\mathbf{x}_n - \mathbf{x}_i][\mathbf{x}_n - \mathbf{x}_i]^T \quad (4.49)$$

$$\bar{\mathbf{P}}_\Delta(\mathbf{w}_n) = (1 - \lambda)\bar{\mathbf{P}}_\Delta(\mathbf{w}_{n-1}) + \frac{\lambda}{L} \sum_{i=n-L}^{n-1} G_{\sigma\sqrt{2}}(e_n - e_i)[z_n - z_i][\mathbf{x}_n - \mathbf{x}_i], \quad (4.50)$$

where the parameters  $\lambda$  and  $L$  are called the forgetting factor and window length. These free design parameters have an effect on the convergence properties of this recursive estimator and have to be established experimentally for the application. The practical importance of this recursive estimator is that it reduces the computational complexity from  $O(n^2)$  in Eqs. (4.47) (4.48) to  $O(L)$ . Depending upon the size of the window, this can be a drastic reduction. Overall, the recursive MEE-FP algorithm of Eq. (4.44) with the updates of Eqs. (4.49) and (4.50) is order  $O(M^3 + M^2L)$ . The  $O(M^2L)$  term is due to the evaluation of the contribution to the RE matrix and PE vector of the current state (input and error). The  $O(M^3)$  term relates to the inversion of the RE matrix.

### Inversion Lemma for MEE-FP

The weight updates by Eq. (4.44) are costly both computationally and memorywise. This is because it requires an inversion of an  $M \times M$  coefficient matrix,  $\bar{\mathbf{R}}_\Delta(\mathbf{w}_n)$  where  $M$  is the filter order, at all time instants. This matrix inversion requires  $O(M^3)$  operations. It is possible to obtain alternative algorithms

based on the matrix inversion lemma [143]. First, convert the summation of the rank-one update Eq. (4.49) to the following

$$\bar{\mathbf{R}}_{\Delta}(\mathbf{w}_n) = (1 - \lambda)\bar{\mathbf{R}}_{\Delta}(\mathbf{w}_{n-1}) + \frac{\lambda}{L}\Pi\Pi^T, \quad (4.51)$$

where  $\Pi = [\mathbf{q}_{n-1}, \dots, \mathbf{q}_{n-L}]$ ,  $\mathbf{q}_i = \sqrt{G_{\sigma\sqrt{2}}(e_n - e_i)}(\mathbf{x}_n - \mathbf{x}_i)$ . Then, apply the matrix inversion lemma

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1} \quad (4.52)$$

with  $A = (1 - \lambda)\bar{\mathbf{R}}_{\Delta}(\mathbf{w}_{n-1})$ ,  $B = (\lambda/L)\Pi$ ,  $C = \mathbf{I}$ , and  $D = \Pi^T$ . We obtain a recursive inverse locally weighted incremental autocorrelation ( $[\bar{\mathbf{R}}_{\Delta}(\mathbf{w}_{n-1})]^{-1} = \bar{\Phi}(\mathbf{w}_n)$ ) as

$$\begin{aligned} \bar{\Phi}(\mathbf{w}_n) &= \frac{1}{1 - \lambda}\bar{\Phi}(\mathbf{w}_{n-1}) - \frac{\lambda}{(1 - \lambda)^2 L}\bar{\Phi}(\mathbf{w}_{n-1})\Pi \\ &\quad \left( \mathbf{I}_{LxL} + \frac{\lambda}{(1 - \lambda)L}\Pi^T\bar{\Phi}(\mathbf{w}_{n-1})\Pi \right)^{-1}\Pi^T\bar{\Phi}(\mathbf{w}_{n-1}). \end{aligned} \quad (4.53)$$

The matrix inversion included in Eq. (4.52) requires  $O(L^3)$  operations. In big- $O$  notation, the complexity of the MEE-FP of Eq. (4.44) with the inversion lemma update of Eqs. (4.51) through (4.53), has a computational complexity of  $O(L^3 + M^2L + ML^2)$ , where the matrix multiplications have complexity  $O(M^2L + ML^2)$ . For small windows this algorithm is preferred. Comparing the recursive estimator for the autocorrelation matrix in the RLS algorithms (e.g., weighted-RLS and extended-RLS [143]) with the MEE-FP counterpart, we again verify the presence of an extra summation due to the pairwise interactions. This term weights the input and desired responses by the Gaussian of the difference in errors. If the errors are similar (remember this similarity is controlled by the kernel size) and  $L = 1$ , the equation defaults to the RLS. This weighting becomes particularly useful for the impulse noise case as we already discussed in Chapter 3.

In terms of computational complexity, whereas in the RLS the estimate of the new autocorrelation is obtained from the current input vector, in the recursive MEE-FP the estimation of the  $R_E$  matrix must account for the relation of the current input and error with all previous inputs and corresponding errors. In practice, however, to limit the computational complexity this estimation is truncated to a window of the last  $L$  samples. Based on this observation it is natural to expect an increase in the computational complexity of the error entropy-based algorithms such as MEE-FP, unless  $L = 1$ .

Comparing the two MEE-FP updates, consider first the situation that  $M$  (filter order) is significantly smaller than  $L$  (window length). In this case the first form of the algorithm is simpler, and the computational complexity simplifies to  $O(M^{-2}L)$  because this is the dominant term. Conversely, if  $L$  is

significantly smaller than  $M$  then the second form of the recursion is more computationally efficient. Similarly, the  $O(M^{-2}L)$  term dominates the computational complexity. Consequently, for a given value of  $M$ , in either case the complexity is higher than that of RLS by a linear factor of  $L$ . This factor is the result of the need for an additional summation in the estimation of the information potential and, in this sense, it is not at all an unreasonable increase. For the extreme case  $L = 1$ , RLS and recursive MEE-FP have the same computational complexity of ( $O(M^2)$ ).

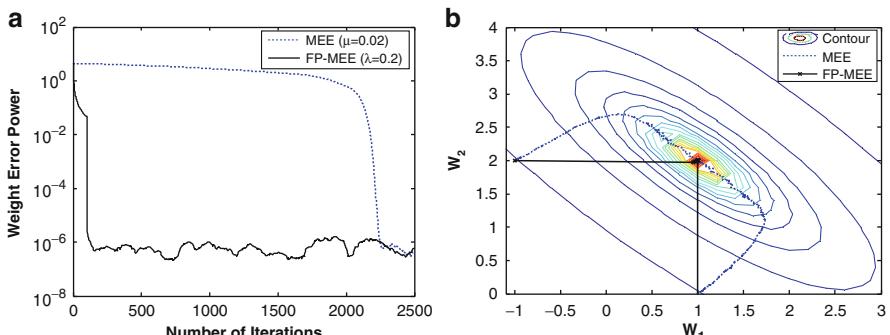
### Effect of Eigenvalue Spread on MEE-FP

In the system identification scenario, both the unknown system and the adaptive system have the same structure. The measurement noise is white Gaussian distributed with zero mean and variance  $10^{-4}$  and is added to the system output. The effect of the eigenvalue spread on both MEE and fixed-point MEE depicted in Figure 4.9 shows the results in the case of colored Gaussian input, whose variance is 30 and a eigenvalue spread ratio ( $S$ ) is 10.

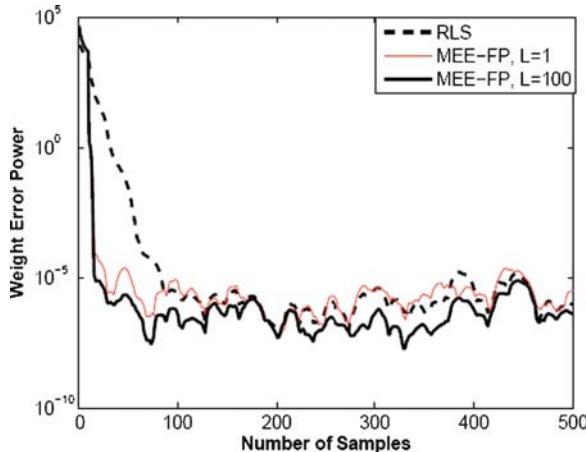
In order to compare the two algorithms, we find the stepsize for the MEE algorithm and the forgetting factor for the fixed-point MEE algorithm to be such that they produce the same misadjustment (around  $10^{-6}$ ). For both algorithms, the kernel size ( $\sigma$ ) is set to 0.707 and the window length  $L$  is set to 100. Also, we fix two initial weight vectors to  $\mathbf{w}_0 = [-1, 2]^T$  and  $[1, 0]^T$ .

The important aspect of these experiments is the different speed and direction of convergence between the MEE and Fixed-Point MEE. From Figure 4.9, we can observe that the fixed-point MEE converges much faster than the MEE algorithm. Figure 4.9b) clearly shows that the weight track of MEE moves along the gradient direction, whereas the fixed-point MEE travels in a direct path to the optimal solution. Therefore, as expected the eigenvalue spread does not affect the speed and direction in the fixed-point MEE algorithm unlike MEE.

One important property experimentally verified is that the FP-MEE still converges well with small windows, in particular  $L = 1$  (Figure 4.10).



**Fig. 4.9.** (a) Average weight error power (b) Weight track on contour of IP surface. Input correlation matrix eigenvalue spread of  $S = 10$ . (from [131]).



**Fig. 4.10.** RLS and FP-MEE convergence for two window sizes  $L = 1$ ,  $L = 100$  and  $S = 10$ . (from [131]).

Figure 4.10 shows the comparison of RLS and FP-MEE (forgetting factor of 0.9 in both) for the two weight system identification problem, and we can observe that although there is a slight increase in misadjustment for  $L = 1$ , the convergence of the FP-MEE for  $L = 1$  is very close to the case  $L = 100$  and better than RLS. The misadjustment of the MEE-FP ( $L = 1$ ) is comparable to the RLS, and the misadjustment of the MEE-FP ( $L = 100$ ) is smaller. For this case, both implementations of MEE-FP are faster than the RLS, perhaps due to the error weighting, but it is unclear if this a general feature of MEE-FP.

## 4.7 Fast Gauss Transform in MEE Adaptation

We have covered two ways to decrease the computational complexity of MEE algorithms: by approximating the IP computation recursively (MEE-RIP) and stochastically (MEE-SIG) which decrease the complexity to  $O(MN)$  and  $O(N)$ , respectively. But the SIG slows down training somewhat due to the noise in the estimate, whereas the RIP is useful in online training of MEE. Here we use the FGT as a means of decreasing the computational complexity of MEE.

The fast Gauss transform (FGT) algorithms presented in Section 2.9 can be applied readily to the MEE algorithm. Suppose that the adaptive system is an FIR structure with a weight vector  $\mathbf{w}$ . The error samples are  $e_k = z_k - \mathbf{w}_k^T \mathbf{x}_k$ , where  $z_k$  is the desired response, and  $\mathbf{x}_k$  is the input vector. For convenience we copy below the MEE gradient equation

$$\nabla \hat{V}(e) = \frac{1}{2\sigma^2 N^2} \sum_{j=1}^N \sum_{i=1}^N G_{\sigma\sqrt{2}}(e_j - e_i)(e_j - e_i)(\mathbf{x}_j - \mathbf{x}_i). \quad (4.54)$$

The FGT with Hermite interpolation can be readily applied to Eq. (4.54) yielding

$$\begin{aligned} \nabla \hat{V}_H(e) = & \frac{1}{2\sigma N^2 \sqrt{\pi}} \sum_{j=1}^N \sum_B \sum_{n=0}^{p-1} \left\{ h_n \left( \frac{e_j - c_B}{2\sigma} \right) \cdot \nabla C_n(B) \right. \\ & \left. + C_n(B) \cdot h_{n+1} \left( \frac{e_j - c_B}{2\sigma} \right) \cdot \left( \frac{\mathbf{x}_j}{2\sigma} \right) \right\}, \end{aligned} \quad (4.55)$$

where  $\nabla C_n(B)$  is defined by

$$\nabla C_n(B) = \frac{1}{n!} \sum_{e_i \in B} n \left( \frac{e_i - c_B}{2\sigma} \right)^{n-1} \cdot \left\{ -\frac{\mathbf{x}_i}{2\sigma} \right\}. \quad (4.56)$$

The FGT with Taylor series expansion is similar and presented in Eq. (4.57)

$$\begin{aligned} \nabla V_T(e) = & \frac{1}{2\sigma N^2 \sqrt{\pi}} \sum_{j=1}^N \sum_B \sum_{n=0}^{p-1} \left[ \exp \left( -\frac{(e_i - c_B)^2}{4\sigma^2} \right) \cdot \left( \frac{e_i - c_B}{2\sigma} \right)^n \right. \\ & \times \nabla C_n(B) - C_n(B) \Bigg], \left[ -2 \cdot \exp \left( -\frac{(e_i - c_B)^2}{4\sigma^2} \right) \cdot \left( \frac{e_i - c_B}{2\sigma} \right)^{n+1} \right. \\ & \left. + n \cdot \exp \left( -\frac{(e_i - c_B)^2}{4\sigma^2} \right) \times \left( \frac{e_i - c_B}{2\sigma} \right)^{n-1} \right] \cdot \left( \frac{\mathbf{x}_i}{2\sigma} \right) \end{aligned} \quad (4.57)$$

where  $\nabla C_n(B)$  is defined by

$$\begin{aligned} \nabla C_n(B) = & \frac{2^n}{n!} \sum_{e_i \in B} - \left\{ -2 \cdot \exp \left( -\frac{(e_i - c_B)^2}{4\sigma^2} \right) \cdot \left( \frac{e_i - c_B}{2\sigma} \right)^{n+1} \right. \\ & \left. + n \cdot \exp \left( -\frac{(e_i - c_B)^2}{4\sigma^2} \right) \cdot \left( \frac{e_i - c_B}{2\sigma} \right)^{n-1} \right\} \cdot \left( \frac{\mathbf{x}_i}{2\sigma} \right). \end{aligned} \quad (4.58)$$

### Entropy Estimation with FGT

We analyzed the accuracy of the FGT in the calculation of the IP for the Gaussian and uniform distributions for two sample sizes (100 and 1000 samples) [131]. For a comparison between SIG and FGT we use  $p = L$  in all our simulations. We fix the radius of the farthest-point clustering algorithm at  $r = \sigma$ . As can be expected, the absolute error between the IP and the FGT estimation decreases with the order  $p$  of the Hermite expansion to very small

values, from  $10^{-4}$  for  $p = 3$  to  $10^{-12}$  for  $p = 20$ , whereas that of the SIG fluctuates around 0.005 (100 samples) and 0.0001 (1000 samples). We can conclude that from a strictly absolute error point of view, a FGT with order  $p > 3$  outperforms the SIG for all cases.

Recall that the Taylor expansion in the FGT saves computation in high dimensions. This is only relevant for the multiple-input-multiple-output (MIMO) case, and indeed we are able to show computational savings, but also a decrease in accuracy when the number of outputs are more than three. Again the results are presented in [131]. For the single-input-single-output (SISO) or MISO cases the Hermite expansion is the preferred approach. However, for our ITL application, the accuracy of the IP is not the only objective. Indeed, in ITL we train adaptive systems using gradient information, so the smoothness of the cost function is perhaps even more important.

## Adaptive Learning with FGT

We consider the system identification of a moving-average plant with a ninth order transfer function given by

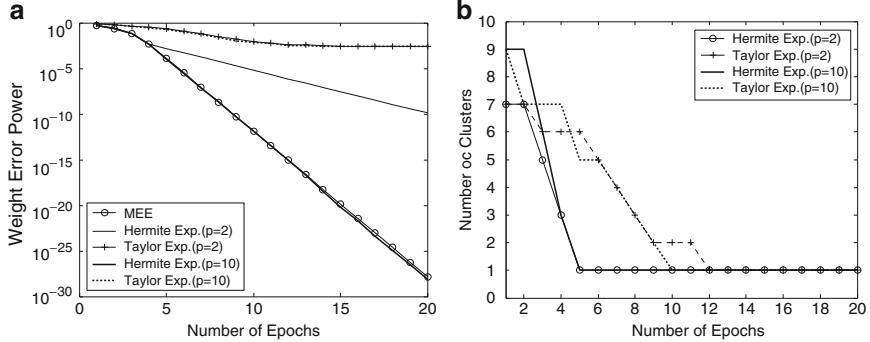
$$H(z) = 0.1 + 0.2z^{-1} + 0.3z^{-2} + 0.4z^{-3} + 0.5z^{-4} + 0.4z^{-5} + 0.3z^{-6} \\ + 0.2z^{-7} + 0.1z^{-8}$$

using the minimization of the error entropy (zero achievable error). In each case the power of the weight noise was plotted versus the number of epochs performed. In this simulation, the inputs to both the plant and the adaptive filter are also white Gaussian noise. We choose a proper kernel size by using Silverman's rule ( $\sigma = 0.707$ ) and fix the radius of the farthest point clustering algorithm  $r = 0.5 \times \sigma$ .

As can be observed in Figure 4.11a, all the adaptive methods of the information potential produce converging filters. However, the speed of convergence and the weight error values of the final epoch are different. The fast method using the Hermite expansion performs better in training the adaptive system as compared to the fast method using the Taylor expansion. The method expanded into a Hermite function with second order has a better performance when compared with that of the tenth order expansion in the Taylor series. The Hermite expansion method with a tenth order expansion is virtually identical to the value of the direct method.

Figure 4.11b shows the plot of the number of clusters during adaptation. Because the error is decreasing at each epoch, the number of clusters gets progressively smaller. In this case, where the achievable error is zero, the number reduces to one cluster after adaptation. The Taylor expansion method with second and tenth order defaults to one cluster after 12 and 10 epochs, respectively, whereas the Hermite expansion method with second and tenth achieves one cluster after 5 epochs.

We have also shown [131] that the fast method using the Taylor expansion, although worse than the Hermite expansion, provides converging filters in



**Fig. 4.11.** (a) Comparison of the direct method and the fast methods (Hermite and Taylor expansion) for system identification with the white uniform noise using  $N (= 2500)$  samples. (b) Plot of the number of clusters during adaptation in system identification. (from [131]).

high dimensions with a small penalty in converging speed (not shown here). Therefore, our recommendation is to use the Hermite approximation for small dimensionality outputs ( $< 3$ ), and the Taylor method in higher dimensions.

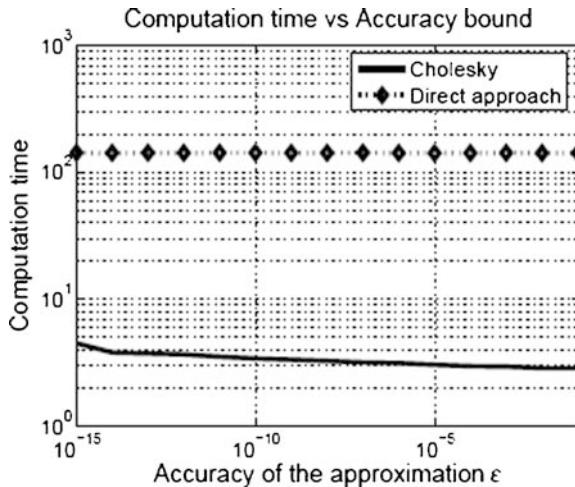
## 4.8 Incomplete Cholesky Decomposition for MEE

A first analysis of the gradient equation for the MEE (Eq. (3.16)) shows that the kernels are multiplied by products of incremental errors and incremental inputs, which in a matrix implementation destroys the Toeplitz structure required to apply the incomplete Cholesky decomposition [116]. However, we can still apply this technique if we rewrite Eq. (3.16) in a slightly different form. In fact, note that the Hadamard product of two matrices  $\mathbf{C}$  and  $\mathbf{A} = \mathbf{ab}^T$ , where  $\mathbf{a}$  and  $\mathbf{b}$  are column vectors, can be written as  $\mathbf{C} \circ \mathbf{A} = \text{diag}(\mathbf{a})\mathbf{C}\text{diag}(\mathbf{b})$  where  $\text{diag}(\mathbf{a})$  is a diagonal matrix with the entries of the  $\mathbf{a}$  vector in the diagonal. With this notation Eq. (3.16) can be rewritten as

$$\nabla_k V(n) = \frac{1}{2N^2\sigma^2} \mathbf{1}^T (\mathbf{K} \circ \Delta \mathbf{E} \circ \Delta \mathbf{X}^{(k)}) \mathbf{1}, \quad (4.59)$$

where matrix  $\mathbf{K}$  has entries  $K_{ij} = G(e(n-i) - e(n-j))$  and the matrix  $\Delta \mathbf{E}$  entries  $\Delta E_{ij} = e(n-i) - e(n-j)$ , as well as  $\Delta \mathbf{X}^{(k)}$  with entries  $\Delta X_{ij}^{(k)} = x_k(n-i) - x_k(n-j)$ . We can further decompose  $\Delta \mathbf{E} = \mathbf{e} \mathbf{1}^T - \mathbf{1} \mathbf{e}^T$  and  $\Delta \mathbf{X}^{(k)} = \mathbf{x}_k \mathbf{1}^T - \mathbf{1} \mathbf{x}_k^T$ . If we apply the identity for the Hadamard product of matrices above in Eq. (4.59), we get

$$\begin{aligned} \nabla_k V(n) &= \frac{1}{2N^2\sigma^2} \mathbf{1}^T (\text{diag}(\mathbf{x}_k)(\text{diag}(\mathbf{e})\mathbf{K} - \mathbf{K}\text{diag}(\mathbf{e}))) \\ &\quad - (\text{diag}(\mathbf{e})\mathbf{K} - \mathbf{K}\text{diag}(\mathbf{e}))\text{diag}(\mathbf{x}_k) \mathbf{1} \end{aligned}$$



**Fig. 4.12.** Computation time for the adaptive filter based on the minimum error entropy criterion. Averages are taken from 15 trials of the computation time of the low rank approximation.

Now if the incomplete Cholesky decomposition is applied to  $\mathbf{K} \sim \Gamma\Gamma^T$ , we obtain

$$\nabla_k V(n) \approx \frac{1}{N^2\sigma^2} (\mathbf{e}^T \Gamma\Gamma^T \mathbf{x}_k - (\mathbf{e} \circ \mathbf{x}_k)^T \Gamma\Gamma^T) \mathbf{1}. \quad (4.60)$$

Equation (4.60) can be computed in  $O(Np)$ , where  $p$  is the number of columns of  $\Gamma$ , instead of  $O(N^2)$ , so the savings are substantial.

We tested the incomplete Cholesky decomposition (ICD) in the adaptation of the same filter as in Section 4.7. Figure 4.12 depicts the behavior of the computation time for different accuracy levels in the ICD. Each trial uses 4000 samples for training and the estimates of the IP gradient are computed using batches of 200 points. Training always converges to the same zero error solution. It is remarkable that even for very strict levels of accuracy the computation time remains about 100 times lower than direct computation. The main reason behind this phenomenon is due to the shrinkage of the error. Each step decreases the entropy of the error distribution resulting in a low-rank Gram matrix that can be efficiently computed by ICD.

## 4.9 Linear Filter Adaptation with MSE, MEE and MCC

We have been applying in this chapter the EEC to linear adaptive filters with no noise in the desired response, and the reader may be induced to think that we suggest such an application instead of the more traditional mean square error cost function. This is not the general case, and our only motivation has

been to apply the MEE to the simplest filter topology where adaption is well understood. The true advantage of the EEC (and ECC) over MSE is when the error distribution is non-Gaussian, which occurs in basically two scenarios: the filter topology is nonlinear or the noise in the desired signal (or the desired signal itself) is non-Gaussian. The former is the object of the next chapter and the latter is exemplified in this section.

We will also include the MCC algorithm in our comparisons because of its close relation with the MEE, and the simpler computational complexity which is linear with the data samples per weight as the LMS algorithm. In fact, Eq. (3.36) presents the ECC and adapting this equation to the Gaussian kernel and the online cost function of adaptive filtering we obtain the MCC cost function as

$$J(n) = \frac{1}{L\sqrt{2\pi}\sigma} \sum_{i=n-L+1}^n e^{-(z(i)-y(i))^2/2\sigma^2} \quad (4.61)$$

for a window size of  $L$  samples and where  $z$  is the desired signal and  $y$  the filter output. Taking the gradient of  $J(n)$  with respect to the filter weight vector  $\mathbf{w}$ , and using the gradient ascent approach for each weight  $w_k$ ,  $w_k(n+1) = w_k(n) + \eta \nabla_k J(n)$  we obtain the MCC steepest gradient ascent update formula as

$$w_k(n+1) = w_k(n) + \frac{\eta}{L\sqrt{2\pi}\sigma^3} \sum_{i=n-L+1}^n e^{-e(i)^2/2\sigma^2} e(i)x_k(i). \quad (4.62)$$

Notice that the stochastic gradient approximation to Eq. (4.62), the SIG(1) only uses the current sample because there is only one element in the sum, yielding

$$w_k(n+1) = w_k(n) + \frac{\eta}{\sigma^2} G_\sigma(e(n))e(n)x_k(n) \quad (4.63)$$

which is the MCC-SIG(1) algorithm with the same complexity as the LMS algorithm. However, now the product error and input are weighted by a non-linear function of the error for Gaussian kernels, so there is an extra control of the weight update using the ECC when compared with MSE. Therefore the ECC cost function yields a stochastic weight update of the same complexity of the LMS, and with some of the properties of the MEE as demonstrated in Chapter 3.

## System Identification in Non-Gaussian Noise Environments

A system identification problem involves determining the coefficients of an unknown plant, by studying its response to a white input signal. The unknown plant is modeled by the adaptive filter in such a way that for the same input, the difference between the outputs of the plant and the filter is minimized.

This adaptation is complicated by the presence of measurement noise which gets added to the output of the plant. A practical extension of the system identification problem is when the transfer function of the plant is changing with time and is required to be tracked by the adaptive filter. For this simulation [302], the input to the plant and the adaptive filter is a white Gaussian signal with zero-mean and unit variance. The coefficients of the plant are again chosen as

$$\mathbf{w}^* = [0.1, 0.2, 0.3, 0.4, 0.5, 0.4, 0.3, 0.2, 0.1]^T.$$

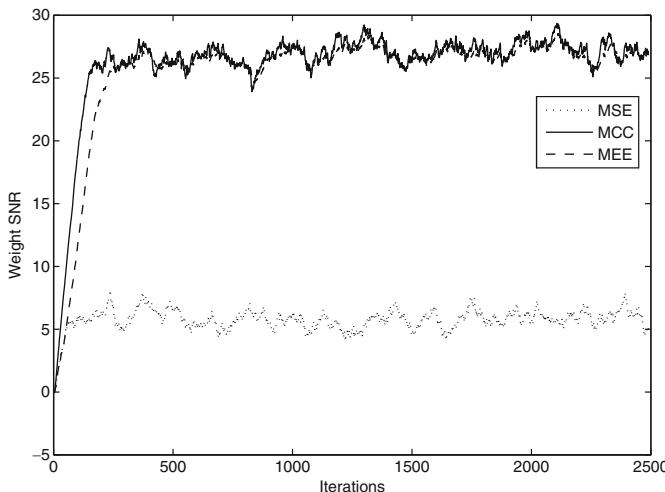
We intend to compare the three cost functions in the presence of an impulsive observation noise, which can be simulated using a mixture of Gaussians:

$$0.95G(0, 10^{-4}) + 0.05G(0, 10),$$

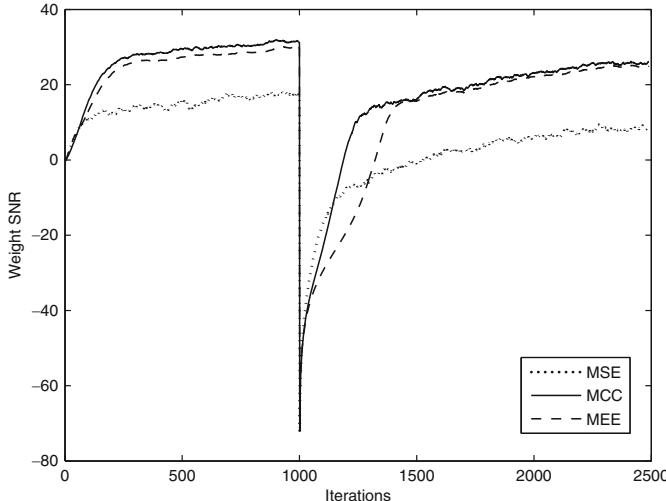
where  $G(m, \sigma)$  is the Normal distribution with mean  $m$  and standard deviation  $\sigma$ . Clearly, in this density, the Gaussian distribution with variance 10 creates strong outliers. The kernel sizes for the MCC and the MEE algorithms are set to 2 for this case. The stepsizes for the three update rules are chosen such that when the observation noise is Gaussian, their performance is similar in terms of the weight SNR defined as,

$$WSNR = 10 \log \left( \frac{\mathbf{w}^{*T} \mathbf{w}^*}{(\mathbf{w}^* - \mathbf{w}(n))^T (\mathbf{w}^* - \mathbf{w}(n))} \right).$$

When adaptation is done in the presence of impulsive noise, we can see a significant difference in performance (Figure 4.13). Whenever a high-amplitude outlier is encountered in the desired signal it is transmitted to the



**Fig. 4.13.** Weight SNR of MSE, MEE, and MCC in impulsive measurements (from [302]).



**Fig. 4.14.** Weight SNR of LMS, MEE, and MCC while tracking a time-varying system in the presence of impulsive measurement noise (from [302]).

error and the LMS weight update rule is forced to make a large increment, which takes the weights away from the true values. The overall effect of several such outliers is that the weights keep jittering around the optimal values, and exhibit noisy weight tracks.

This can be seen from the weight SNR (WSNR) plots in Figure 4.13. The MCC weight tracks are more robust in such situations. Referring back to the MCC update rule, Eq. (4.62), the exponential function of the error provides the attenuation when high-amplitude outliers are encountered, keeping the overall weight track close to the optimal. The MEE criterion also exhibits similar characteristics in the presence of impulsive noise for a similar reason (see Eq. (4.15)).

We now simulate a time-varying plant transfer function where the coefficients are changing as follows,

$$\mathbf{w}^*(n) = 2 \left(1 + \frac{n}{1000}\right) u(1000 - n) \mathbf{w}^* + \left(-1 + \frac{n}{1000}\right) u(n - 1000) \mathbf{w}^*,$$

where  $u(n)$  is the unit step function. Figure 4.14 shows the results of tracking the above plant weights. The performance of the MCC algorithm is better in this case. As compared to the MEE algorithm, it is able to track the weights better as can be seen from the portion of the curves just after the discontinuity ( $n = 1000$ ).

### Adaptive Noise Cancellation of Speech Signals

An important application of adaptive filters is in acoustic noise cancellation. Given a desired signal (speech, music, etc.) corrupted by ambient noise, an

adaptive filter tries to replicate this noise by exactly modeling the acoustic transfer function between the noise source and the signal. In practical scenarios, both the noise and the acoustic transfer function may be nonstationary.

Figure 4.15 shows the configuration of a noise cancellation system. The difference between the desired signal  $z$  and the output of the filter  $y$  is in fact the noise-free signal (cleaned speech), which is a non-Gaussian signal (positive kurtosis, also called super-Gaussian). We have tried to simulate real-life conditions as closely as possible [302]. The speech signal is shown in Figure 4.16a. The noise signal is that of a vacuum cleaner collected in a room, and it is nonstationary as can be seen in Figure 4.16b. The acoustic transfer function is that of a typical closed room environment. We use a 200-tap filter to model the acoustic path.

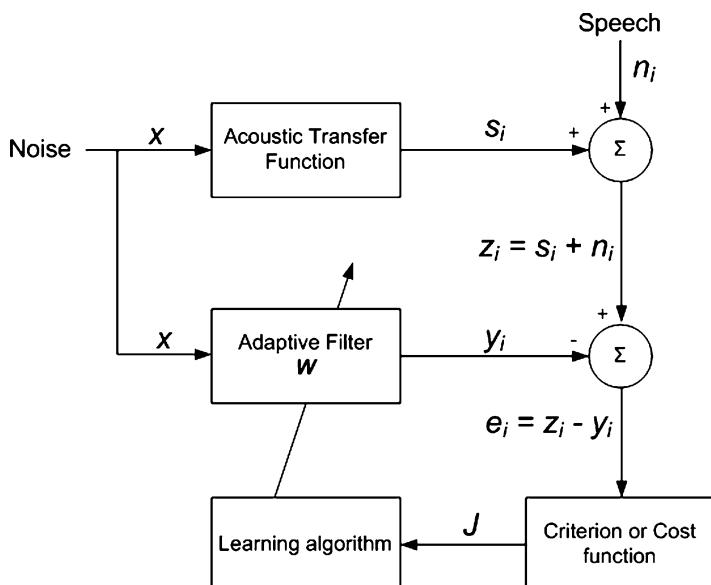


Fig. 4.15. An acoustic noise cancellation configuration.

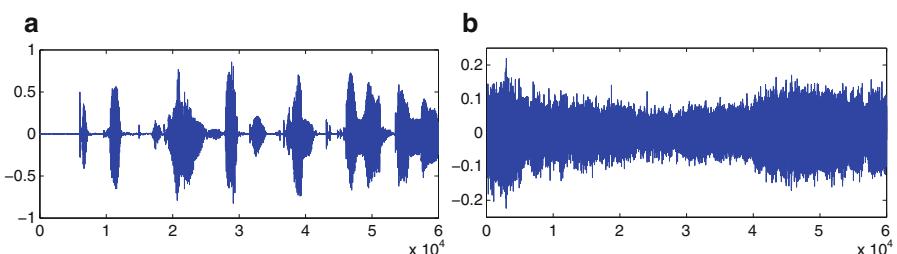
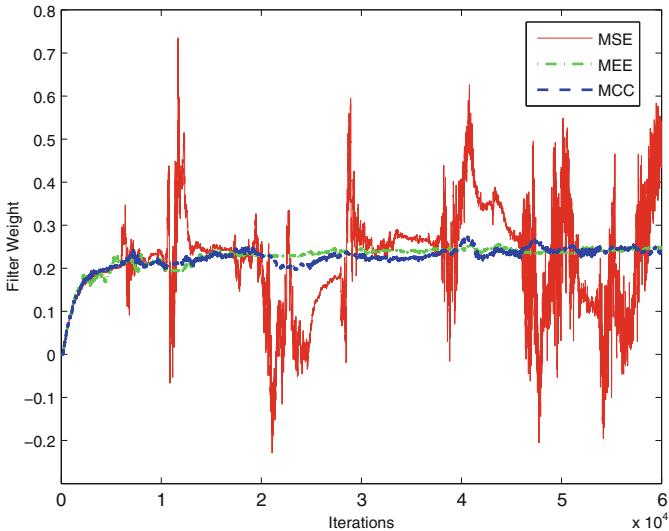


Fig. 4.16. (a) Speech recording and (b) vacuum cleaner noise.



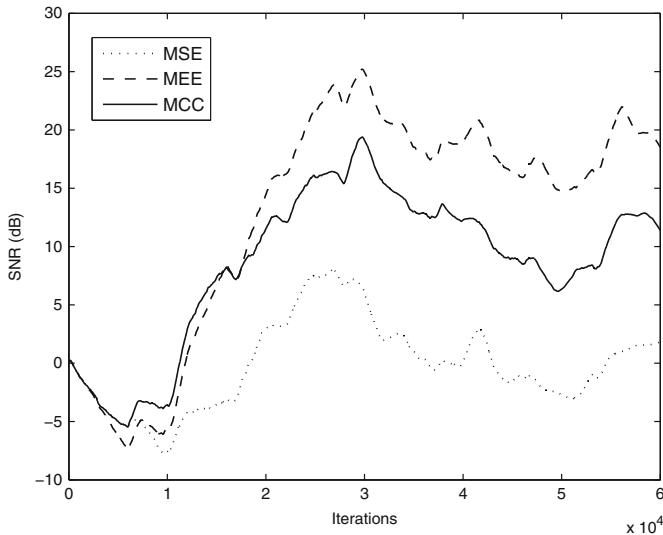
**Fig. 4.17.** Weight tracks for one of the filter weights. The convergence time for all the methods has been kept the same. The LMS procedure is much less stable at steady state in such conditions.

The parameters of the weight update equations were chosen to have a high rate of convergence, and in all three cases were set to be the same. The MCC algorithm was found to be quite robust in the presence of sudden and irregular “bursts” that appear in the voiced portions of speech. Figure 4.17 shows the weight tracks of one of the filter weights.

For the same convergence rate, the MSE cost function produced a much higher misadjustment at steady state. This is because of the highly impulsive nature of the speech signal. The sudden, high-amplitude “bursts” that occur in speech signals can easily disturb the LMS weight track. However, correntropy, being a localized similarity measure, places exponentially decreasing weights on samples that are distant and impulsive. Therefore, the overall weight tracks of filters trained with MCC are much less affected by such types of nonstationarities in the signal.

Figure 4.18 shows the signal-to-noise ratio between the original speech signal and the cleaned signal (after averaging over a moving window of 10,000 samples, for visual clarity). There is a substantial improvement in the SNR as obtained by MEE when compared with the MSE criterion. The MCC SNR is also much better than LMS but slightly worse than MEE, but it is important to remember that the MCC results are obtained at a much simpler computational complexity as compared to the MEE filter.

Echo return loss enhancement (ERLE) is a commonly used measure of performance for echo cancellers. It measures the amount of noise (or echo) power that has been removed from the desired signal by subtracting the output



**Fig. 4.18.** SNR (smoothed over a running window of 10,000 samples) obtained after noise cancellation with LMS, MCC, and MEE cost functions.

of the adaptive filter. It is a ratio of the power of the desired signal and the error, expressed in decibels:

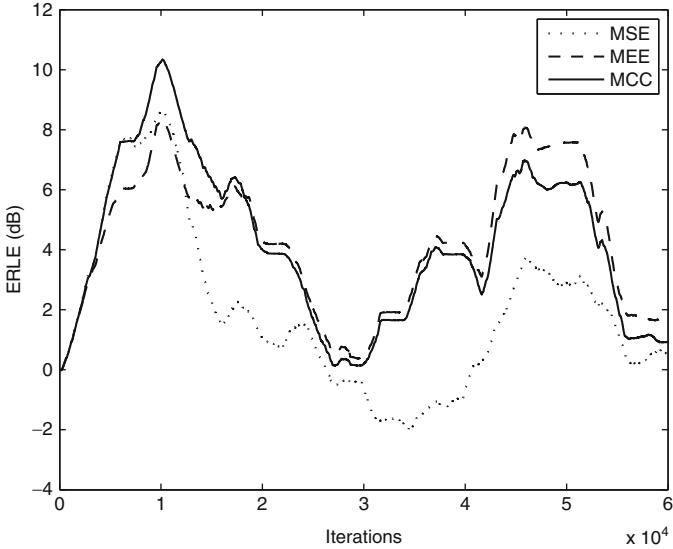
$$ERLE(\text{dB}) = 10 \log \left( \frac{E[z_i^2]}{E[e_i^2]} \right).$$

Figure 4.19 shows the ERLE values of the three techniques, after smoothing using a running window of 10,000 samples, for visual inspection. Clearly, the MEE and MCC algorithms are able to remove more noise power from the desired signal, as compared to the LMS.

From these preliminary tests we can say that for real-world applications requiring low computational complexity such as for portable devices, the MCC is a very strong contender. Of course the kernel size must be determined from the data, unless the adaptive kernel size algorithm of Section 3.7 is used. In this case no extra parameters when compared with the conventional LMS are required, but the adaptation of  $\sigma$  is  $O(N)$ .

## 4.10 Conclusion

This chapter explained in great detail a set of algorithms for the error entropy criterion called minimum error entropy that can be applied to both adaptive filters or in regression. Conceptually EEC is a more powerful cost function because it deals directly with the PDF of the error. During adaptation the



**Fig. 4.19.** ERLE values (smoothed over a running window of 10,000 samples) obtained after noise cancellation with LMS, MCC, and MEE cost functions.

PDF of the error changes, therefore the weights of the filter control the shape of the PDF.

The use of linear FIR of small order was dictated by easy visualization of the solution. In fact, we do not recommend the use of EEC with FIR filters when signals are stationary in noise free environments, which normally leads to Gaussian residuals (recall that the FIR adds random variables, so the filter output tends to be Gaussian). But again the linear FIR case is the simplest of the architectures so insights of how EEC works in nonlinear systems can be inferred from this chapter.

We have presented several different adaptation algorithms for the EEC that parallel most of the gradient descent methods (and RLS) for MSE. Some of the MEE algorithms (MEE-RIP, MEE-SIG) attempt to decrease the computation of the gradient (i.e.,  $O(N^2)$  if nothing is done). We propose a recursive estimation of the IP and a stochastic approximation that drop the computation to  $O(L)$ , where  $L$  is the number of samples in the window used to estimate the IP. We also addressed the difficulty of searching the EEC using the MEE-SAS (self-adjusting stepsize) that speeds up the adaptation considerably because it has a fourth order convergence in a large portion of the space (very similar to the LMF algorithm). There is also an NMEE algorithm that alleviates the problem of large swings in the amplitude power, and also reduces the dependency of the kernel size in the search. We also presented a second-order search method for the MEE based on a fixed-point update, which is insensitive to eigenvalue spread, but is more computationally costly.

Finally, we applied the fast Gauss transform approximation to the training of MEE, and found out that small order expansions ( $p < 10$ ) and few clusters are sufficient to create an accurate search of the EEC performance surface, and they considerably decrease the computational complexity. The Taylor series expansion should be used in higher dimensions for efficiency in the calculations, but it is less accurate. Regarding the incomplete Cholesky decomposition explained in Chapter 2 as an alternative to speed up computation, we show how it can still be applied to gradient computation in block form (i.e., for the batch mode algorithms). The ICD has advantages because it is a well known matrix manipulation so it is simpler to apply than the FGT. It is also able to decrease the computational accuracy product 100-fold, effectively yielding  $O(N)$  algorithms, which means that it is a practical methodology when simulating or implementing EEC solutions using the IP estimator. Therefore, we consider that a sufficiently rich set of algorithms exists to apply EEC in practical applications. However, not all is known about this new cost function. The selection of the kernel size is still an open problem (minimized by the NMEE), as well as the determination of the learning rate. They both need further research and it may be possible to derive an adaptive kernel size algorithm (see [302] for some preliminary results).

The chapter ends with a comparison among the MSE, EEC, and ECC cost functions under the linear model to illustrate the fact that even in the simple scenario of linear models there are advantages of using the MEE and MCC algorithms over LMS when the noise is non-Gaussian or when the signals of interest are non Gaussian as in the case of acoustic noise cancellation. The MCC has a great advantage with respect to the MEE in terms of computational complexity, because a single sum is necessary to evaluate the cost. In practical problems, the MCC has performance very close to the MEE, therefore its applicability in low-power, portable DSP processors may be preferable to MEE. Although not yet fully studied, the class of error entropy and error correntropy cost functions may provide faster convergence for the same misadjustment because the cost functions are not constant curvature as is the MSE cost. The issue is how to choose the operating point to achieve faster convergence as initiated in the MEE-SAS approach.

## Nonlinear Adaptive Filtering with MEE, MCC, and Applications

Deniz Erdogmus, Rodney Morejon, and Weifeng Liu

### 5.1 Introduction

Our emphasis on the linear model in Chapter 4 was only motivated by simplicity and pedagogy. As we have demonstrated in the simple case studies, under the linearity and Gaussianity conditions, the final solution of the MEE algorithms was basically equivalent to the solution obtained with the LMS. Because the LMS algorithm is computationally simpler and better understood, there is really no advantage to use MEE in such cases.

The primary domain where the EEC criterion will likely provide advantages with respect to MSE is when the cost function works on a random variable that is not Gaussian distributed. The case where the observation noise added to the desired response, or the desired response itself is non-Gaussian, was already exemplified in Chapter 4 with speech echo noise cancellation (the speech becomes the error). Here we study the nonlinear systems case, where even if the input–desired response pairs are Gaussian distributed random variables the error is likely not Gaussian distributed due to system nonlinearities. These are two already identified conditions where the EEC may provide superior performance, but others may still be discovered due to the differences in the EEC, ECC and MSE cost functions.

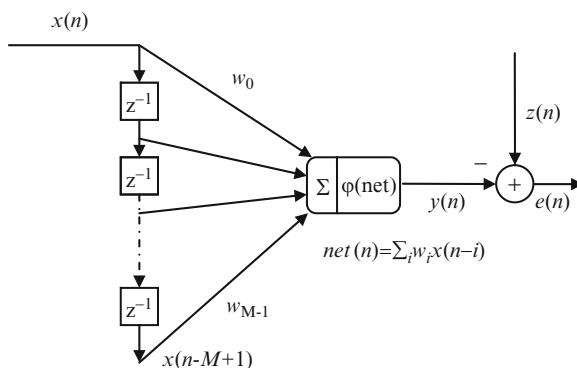
We show that the MEE algorithms can be easily extended to nonlinear adaptive filtering by integrating the error entropy cost function with the back-propagation algorithm (BP) of neural network theory [253]. In as much as for nonlinear systems the cost functions become nonconvex, it is important to go beyond the conventional gradient descent procedure explained in Chapter 4. We address the modifications made to advanced search procedures when ITL cost functions are utilized. Finally, this chapter also covers several important applications of information filtering to engineering applications to illustrate the gains and the important parameters that need to be controlled to achieve operational performances close to the optimum EEC can provide.

## Nonlinear Adaptive Filtering with Error Entropy Criterion

Neural networks are an important engineering tool in nonlinear signal processing, therefore this chapter addresses the application of the EEC and its MEE algorithm to neural networks. The fundamental question is: what changes in the large body of neural network training methods when the MSE is substituted by the EEC? The answer is rather simple. If we carefully examine together the gradient equations for the MSE and EEC (Eqs. (1.33) and (3.15)) using the chain rule across the system topology we see that the effect of the EEC appears encapsulated in the partial derivative of the cost with respect to the error ( $\partial J/\partial e$ ), which means that when we switch between MSE and EEC this is the only part of the gradient computation that changes. The other aspect that must be noticed in Eq. (3.15) is that the MEE algorithm works with pairs of samples; that is, the gradient is no longer local in time, but it can still be written compactly as differences in gradients. The propagation of the sensitivities throughout the topology ( $\partial e/\partial w$ ) is computed from the partial derivative of the error with respect to the system parameters. Therefore, when a nonlinear topology is selected instead of a linear one, the propagation of sensitivities is where most changes occur, which is a topic well treated in the theory of neural network training [253]. We can thus expect that the MEE algorithm can be easily integrated in neural network training. Let us basically explain the procedure with the simplest nonlinear topology of Figure 5.1 which is called a focused time delay neural network (TDNN) with a single layer and one single processing element (PE) for simplicity.

In the figure  $\varphi(\cdot)$  is a static differentiable nonlinearity, normally a sigmoid [253]. The sensitivity of the cost  $J$  at time  $n$  with respect to the weight  $w_k$  can be computed using the ordered derivative over the topology, to read

$$\frac{\partial J}{\partial w_k} = \frac{\partial J}{\partial e(n)} \frac{\partial e(n)}{\partial \text{net}(n)} \frac{\partial \text{net}(n)}{\partial w_k}. \quad (5.1)$$



**Fig. 5.1.** A simple TDNN network.

Notice that when compared with Eq. (1.33) we now have a third partial derivative (the middle term) in the chain rule applied to the topology that relates the error and the argument of the nonlinearity which we called  $\text{net}(n)$ . If we use the MSE cost, Eq. (5.1) immediately gives

$$\frac{\partial J}{\partial w_k} = -2E(e(n))\varphi'(\text{net}(n))x_k(n), \quad (5.2)$$

where  $\varphi'(\text{net})$  stands for the derivative of the nonlinearity evaluated at the operating point  $\text{net}(n)$ . We just rederived the delta rule of neural network learning [253].

In the case of the MEE procedure, recall that  $\hat{V}(E) = 1/N^2 \sum_{i=1}^N \sum_{j=1}^N \kappa(e(i) - e(j))$ , with an arbitrary kernel  $\kappa$ , so the counterpart of Eq. (5.1) is

$$\begin{aligned} \frac{\partial \hat{V}(E_n)}{\partial w_k} &= \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \frac{\partial \kappa(e(n-i) - e(n-j))}{\partial (e(n-i) - e(n-j))} \frac{\partial (e(n-i) - e(n-j))}{\partial w_k} \\ &= \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \frac{\partial \kappa(e(n-i) - e(n-j))}{\partial (e(n-i) - e(n-j))} \\ &\quad \left( \frac{\partial e(n-i)}{\partial \text{net}(n-i)} \frac{\partial \text{net}(n-i)}{\partial w_k} - \frac{\partial e(n-j)}{\partial \text{net}(n-j)} \frac{\partial \text{net}(n-j)}{\partial w_k} \right). \end{aligned} \quad (5.3)$$

The big difference is simply in the use of pairs of samples, which means that the gradient has to be computed for each sample in the pair, over all pairs. When evaluated with Gaussian kernels Eq. (5.3) yields the batch gradient estimate for the MEE delta-rule

$$\begin{aligned} \frac{\partial \hat{V}(E_n)}{\partial w_k} &= \frac{1}{2\sigma^2 N^2} \sum_{i=1}^N \sum_{j=1}^N G_{\sigma\sqrt{2}}(e(n-i) - e(n-j))(e(n-i) - e(n-j)) \\ &\quad (\varphi'(\text{net}(n-i))x_k(n-i) - \varphi'(\text{net}(n-j))x_k(n-j)), \end{aligned} \quad (5.4)$$

where  $\varphi'$  is the derivative of the PE nonlinearity. The interpretation of information force given in Chapters 2 and 3 about the first component of the gradient in Eq. (5.3) still applies here, but now notice that the derivative of the nonlinearity evaluated at the operating point is weighting the information force (just as it was modulating the error in the MSE cost function). Of course the nonlinear topology we utilized is very simple, but the principle can be easily generalized for topologies with hidden processing elements and the general procedure is called the backpropagation algorithm [277, 331].

## 5.2 Backpropagation of Information Forces in MLP Training

In this section, we derive the backpropagation algorithm for a multilayer perceptron (MLP) trained under the MEE algorithm, which is called the MEE-BP, that is the adaptation of the MLP weights using a cost function of

the form of Eq. (3.12). This extended algorithm backpropagates the information forces between the error samples in the output error space through the layers of the MLP instead of the error in the standard MSE criterion case. For simplicity, consider the unweighted total potential of the error samples as the cost function. Assume that for multi output situations, we simply sum the potentials of the error signals from each output (as explained in Section 4.1).

Consider an MLP that has  $L$  layers with  $m_l$  PEs in the  $l$ th layer. We denote the input vector with layer index 0. Let  $w_{ji}^l$  be the weight connecting the  $i$ th PE of layer  $l - 1$  to the  $j$ th PE in the  $l$ th layer. Let  $\hat{V}_j^l(s)$  be the information potential of the  $j$ th PE at the  $l$ th layer corresponding to the input sample  $x(s)$ , where  $s$  is the sample index. Assume further that we have  $m_L$  output PEs. Let  $\varphi(\cdot)$  be the sigmoid nonlinearity of the MLP, the same for all PEs, including the output layer. Assume we have  $N$  training samples. The information potential of the system of error samples, given by  $\{e(1), \dots, e(N)\}$ , is then

$$\hat{V} = \sum_{s=1}^N \sum_{t=1}^N \sum_{k=1}^{m_L} \hat{V}(e_k(s) - e_k(t)) \triangleq \sum_{s=1}^N \sum_{t=1}^N \varepsilon(s, t). \quad (5.5)$$

The derivation of the *backpropagation of information forces* algorithm follows along similar lines to those of the conventional error backpropagation algorithm [253]. The total information potential of the output errors summed over the output PEs, for a given sample pair  $(s, t)$  is defined as

$$\varepsilon(s, t) \triangleq \sum_{k=1}^{m_L} \hat{V}(e_k(s) - e_k(t)). \quad (5.6)$$

For this MLP, the output  $y$  of the  $k$ th PE before and after the nonlinearity of the  $l$ th layer is respectively given by

$$net_k^l = \sum_{i=0}^{m_{l-1}} w_{ki}^l y_i^{l-1} \quad y_k^l = \varphi(net_k^l). \quad (5.7)$$

Taking the derivative of  $\varepsilon(s, t)$  with respect to the output layer weights, we obtain Eq. (5.8), where  $\varphi'(\cdot)$  is the derivative of the MLP's sigmoid function and  $\zeta_k^l(\cdot, \cdot)$  are the sensitivities of the local information potentials in the network that depend on the information forces between the indicated samples, which will be denoted by  $\hat{F}(s, t) = \hat{V}'(e(s) - e(t))$ .

$$\begin{aligned} \frac{\partial \varepsilon(s, t)}{\partial w_{ki}^l} &= \hat{V}'(e(s) - e(t)) \cdot [-\varphi'(net_k^l(s))y_i^{l-1}(s) + \varphi'(net_k^l(t))y_i^{l-1}(t)] \\ &= \hat{V}'(e(s) - e(t))\varphi'(net_k^l(t))y_i^{l-1}(t) - \hat{V}'(e(s) - e(t))\varphi'(net_k^l(s))y_i^{l-1}(s) \\ &\triangleq \zeta_k^l(t, s)y_i^{l-1}(t) + \zeta_k^l(s, t)y_i^{l-1}(s). \end{aligned} \quad (5.8)$$

For the hidden layer  $l - 1$ , and for arbitrary weight  $w_{ji}$  we can write, similarly,

$$\begin{aligned}
\frac{\partial \varepsilon(s, t)}{\partial w_{ji}^{l-1}} &= \hat{V}'(e(s) - e(t)) \left[ \begin{array}{c} \frac{\partial e(s)}{\partial y_j^{l-1}(s)} \frac{\partial y_j^{l-1}(s)}{\partial \text{net}_j^{l-1}(s)} \frac{\partial \text{net}_j^{l-1}(s)}{\partial w_{ji}^{l-1}} \\ - \frac{\partial e(t)}{\partial y_j^{l-1}(t)} \frac{\partial y_j^{l-1}(t)}{\partial \text{net}_j^{l-1}(t)} \frac{\partial \text{net}_j^{l-1}(t)}{\partial w_{ji}^{l-1}} \end{array} \right] \\
&= \hat{V}'(e(s) - e(t)) \left[ \begin{array}{c} - \sum_{k=1}^{m_l} \varphi'(\text{net}_k^l(s)) w_{kj}^l \varphi'(\text{net}_j^{l-1}(s)) y_i^{l-2}(s) \\ + \sum_{k=1}^{m_l} \varphi'(\text{net}_k^l(t)) w_{kj}^l \varphi'(\text{net}_j^{l-1}(t)) y_i^{l-2}(t) \end{array} \right] \\
&= \sum_{k=1}^{m_l} \zeta_k^l(t, s) w_{kj}^l \varphi'(\text{net}_j^{l-1}(t)) y_i^{l-2}(t) \\
&\quad + \sum_{k=1}^{m_l} \zeta_k^l(s, t) w_{kj}^l \varphi'(\text{net}_j^{l-1}(s)) y_i^{l-2}(s) \\
&\triangleq \zeta_k^{l-1}(t, s) y_i^{l-2}(t) + \zeta_k^{l-1}(s, t) y_i^{l-2}(s). \tag{5.9}
\end{aligned}$$

The sensitivities of the other hidden layers (if there are more than two) can be computed using the same idea, resulting in similar equations. This derivation and the main points of the algorithm can be summarized as follows. In the algorithm below,  $\eta$  is the learning rate.

*MEE-BP Summary.* Let the information force acting on sample  $s$  due to the potential field of sample  $t$  be  $\hat{F}(e_j(s), e_j(t)) = \hat{V}'(e_j(s) - e_j(t))$  in the  $j$ th output node of the MLP. These interactions minimize the IP in Eq. (5.5).

- Evaluate local gradients for the output layer for  $s, t = 1, \dots, N$  and  $j = 1, \dots, m_l$  using

$$\begin{aligned}
\zeta_j^l(s, t) &= \hat{F}(e_j(s), e_j(t)) \cdot \varphi'(\text{net}_j^l(s)) \\
\zeta_j^l(t, s) &= -\hat{F}(e_j(s), e_j(t)) \cdot \varphi'(\text{net}_j^l(t)). \tag{5.10}
\end{aligned}$$

- For layer index  $l = L, \dots, 1$  (decreasing index) evaluate the local gradients

$$\begin{aligned}
\zeta_j^{l-1}(s, t) &= \varphi'(\text{net}_j^{l-1}(s)) \sum_{k=1}^{m_l} \zeta_k^l(s, t) w_{kj}^l \\
\zeta_j^{l-1}(t, s) &= \varphi'(\text{net}_j^{l-1}(t)) \sum_{k=1}^{m_l} \zeta_k^l(t, s) w_{kj}^l. \tag{5.11}
\end{aligned}$$

- Once all the sensitivities  $\zeta$  are computed, update the weight (for gradient descent) by

$$\Delta w_{ji}^l = -\eta (\zeta_j^l(s, t) y_i^{l-1}(s) + \zeta_j^l(t, s) y_i^{l-1}(t)). \tag{5.12}$$

Information potential cost functions in general are insensitive to the mean position of the desired response, therefore, in applications where the mean of the samples is also desired to be zero (i.e. supervised training), an external force acting on all the particles to draw them towards the zero value must be introduced. Since forces are additive, the information force of the last layer sensitivities in Eq. (5.9) must be the superposition of the information force produced by the cost and the external force acting on that sample. Practically, the introduction of a bias at the output processing element set by the mean of the desired response implements this goal, as already discussed in Chapter 3.

### 5.3 Advanced Search Methods for Nonlinear Systems

When adapting nonlinear systems, very often the gradient descent procedure is not the most recommended due to the flat spots and even local minima of the performance surfaces created by the nonlinear topologies. There is a very large class of methods to speed up learning by changing the stepsize adaptively, or even using second-order search procedures that exploit information of the curvature of the performance surface [285]. All these algorithms have been developed with MSE cost functions in mind, so it is important to see how these advanced search methods can be extended to the EEC and other ITL cost functions. 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 [38]. Several of these advanced search methods are briefly described below in anticipation of their application in EEC training or in more general terms with ITL cost functions.

#### Gradient Descent with Momentum

The addition of a momentum term to gradient descent (GD) serves as a low-pass filter for parameter adaptation. This allows parameter adaptation to avoid 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 as,

$$\Delta w_k = (1 - \alpha)(-\eta \nabla_k \mathbf{J}) + \alpha \Delta w_{k-1}, \quad (5.13)$$

where  $\alpha$  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 the adaptive learning rate and momentum (GDX) is studied as well.

#### Resilient Backpropagation

Resilient backpropagation (RP) was introduced to help eliminate some of the problems encountered when training neural networks containing saturating

nonlinearities using GD methods [267]. 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: (1) if a weight change is in the same direction for the last two epochs, the magnitude of the change is increased; (2) if the update direction is different, the magnitude of change is decreased. The update rule for RP is given as

$$\Delta w_{ij}(k) = -\text{sign}(\nabla J_{ij}(k))r_{ij}(k), \quad (5.14)$$

where the magnitude,  $r_{ij}(k)$ , is adapted as

$$r_{ij}(k) = \begin{cases} ar_{ij}(k-1) & \text{if } \nabla J_{ij}(k) * \nabla J_{ij}(k-1) > 0 \quad \text{where } a > 1 \\ br_{ij}(k-1) & \text{if } \nabla J_{ij}(k) * \nabla J_{ij}(k-1) < 0 \quad \text{where } b < 1. \end{cases} \quad (5.15)$$

RP has proven to be a robust algorithm in as much as 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 [285]. Most conjugate gradient methods determine the change magnitude using a line search technique. The basic update for conjugate gradient methods is

$$\Delta \mathbf{w}_k = \alpha_k \mathbf{d}_k \quad (5.16)$$

$$\mathbf{d}_k = -\nabla \mathbf{J}_k + \beta_k \mathbf{d}_{k-1}, \quad (5.17)$$

where  $\alpha_k$  is determined using a line search  $\alpha_k = \min_\lambda J(w_k + \lambda d_k)$ ,  $\mathbf{d}_k$  represents the conjugate search direction, and  $\beta_k$  determines the method by which the next conjugate direction is chosen. The Fletcher-Reeves (CGF) method chooses the direction by

$$\beta_k = \frac{\nabla \mathbf{J}_k^T \nabla \mathbf{J}_k}{\nabla \mathbf{J}_{k-1}^T \nabla \mathbf{J}_{k-1}}, \quad (5.18)$$

whereas the Polak–Ribiere (CGP) updates the direction with,

$$\beta_k = \frac{(\nabla \mathbf{J}_{k-1} - \nabla \mathbf{J}_{k-2})^T \nabla \mathbf{J}_k}{\nabla \mathbf{J}_{k-1}^T \nabla \mathbf{J}_{k-1}}. \quad (5.19)$$

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. 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.

The scaled conjugate gradient (SCG) method was developed to eliminate the computationally expensive line search from the conjugate gradient approach [221]. 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  $\lambda$  to govern the trust region. The basic update takes the form of Eqs. (5.16) and (5.17) with,

$$\alpha_k = -\frac{\mathbf{d}_k^T \nabla \mathbf{J}_k}{\mathbf{d}_k^T \mathbf{H}_k \mathbf{d}_k + \lambda_k \|\mathbf{d}_k\|^2} \quad (5.20)$$

$$\beta_k = \frac{\nabla \mathbf{J}_k^T \nabla \mathbf{J}_k - \nabla \mathbf{J}_k^T \nabla \mathbf{J}_{k-1}}{-\mathbf{d}_{k-1}^T \nabla \mathbf{J}_k}, \quad (5.21)$$

where  $\lambda$  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 ( $\mathbf{H}$ ), to arrive at the performance minimum in fewer steps. The basic update for Newton's method is given by

$$\Delta \mathbf{w} = -\mathbf{H}^{-1} \nabla \mathbf{J}. \quad (5.22)$$

However, because 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 [285]. Quasi-Newton approaches determine the change direction using the inverse Hessian estimate and the change magnitude using a line search as follows,

$$\Delta \mathbf{w} = -\alpha \mathbf{H}^{-1} \nabla \mathbf{J}, \quad (5.23)$$

where  $\alpha$  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 [20]. The method assumes that for each epoch the previous Hessian estimate was the identity matrix and uses the same update rule as Eq. (5.23). Each of these methods assumes a static performance surface for proper execution of their line searches.

### Levenberg–Marquardt Algorithm

The Levenberg–Marquardt (LM) algorithm uses a model-trust-region technique specifically designed to minimize the mean square error (MSE) function. The model assumes a locally linear network, which produces a parabolic error surface [38]. Based on these assumptions, the algorithm estimates the Hessian matrix and gradient as

$$\mathbf{H} = \mathbf{J}^T \mathbf{J} \quad (5.24)$$

$$\nabla \mathbf{J} = \mathbf{J}^T \mathbf{e}, \quad (5.25)$$

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

$$\Delta w = - \left[ \mathbf{J}^T \mathbf{J} + \mu \mathbf{I} \right]^{-1} \mathbf{J}^T \mathbf{e}. \quad (5.26)$$

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 Assessment Index

Some of the above algorithms use a performance assessment index 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 as

$$J_{k+1} > J_k. \quad (5.27)$$

If the test in Eq. (5.27) is false, it is considered positive for the goals of learning and it is considered against the learning goal when true. However,

in order 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 criteria this is typically implemented by allowing a five percent degradation tolerance,

$$J_{k+1}/J_k > 1.05. \quad (5.28)$$

If Eq. (5.28) evaluates to true, the performance is considered degraded beyond the tolerable limit, the current update is discarded, and the search parameters are adjusted to stabilize learning accordingly. If it evaluates as false, adaptation is continued using either the current search parameters or parameters adjusted to speed up learning.

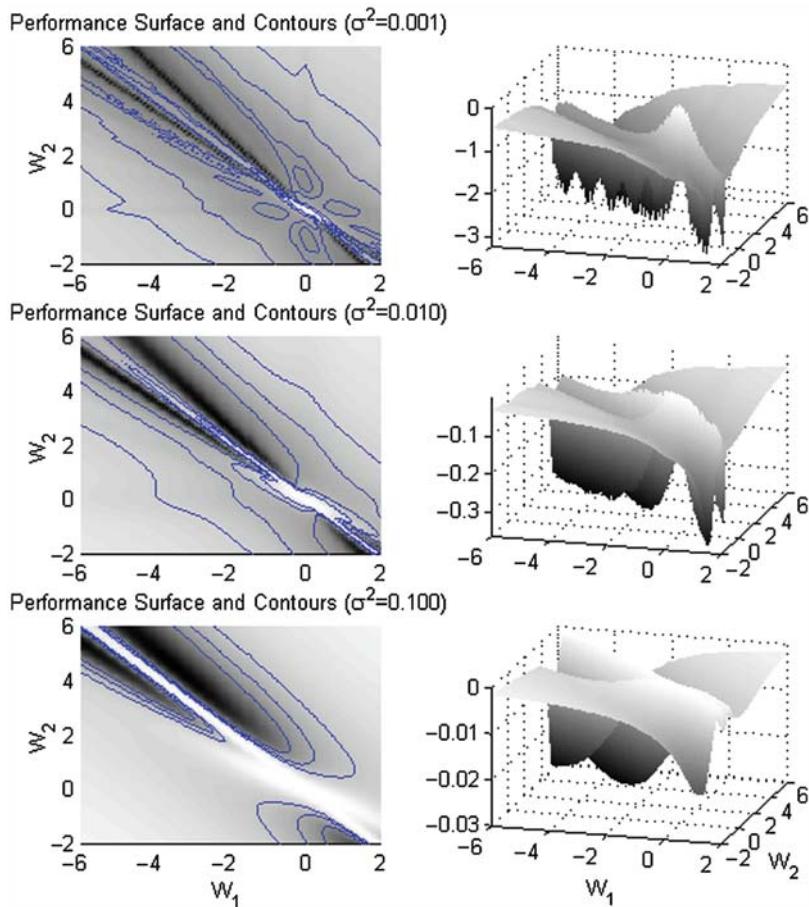
## 5.4 ITL Advanced Search Algorithms

These algorithms have been developed and adapted to the MSE cost function, therefore some of their assumptions may interfere with the peculiarities of ITL algorithms. In general, application of the advanced parameter search techniques described above to EEC and ITL in general (meaning entropy and divergence costs) are straightforward, but care must be taken with the specificities of the new costs for acceptable results. This section concludes with a summary of how the various advanced training algorithms mentioned above have been adapted for use with ITL [224].

### Adaptive Kernel Size

Both entropy and divergence costs of ITL have a free parameter, the kernel size, which is at the core of the learning process because adaptation is caused by interactions between samples that are mediated by their spacing and the kernel size. Research has demonstrated that allowing the kernel size to adapt during training can lead to improved results as mentioned in Chapter 3. However, straight application of the adaptive kernel size in ITL can present problems for many of the advanced parameter search techniques. The cause of the problems is that the kernel size controls the shape of the performance surface. Figure 5.2 compares the performance surface for three different values of kernel size ( $\sigma^2 = 0.001, 0.01, 0.1$ ) for the frequency double example presented next.

The figure illustrates 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 in the case when  $\sigma^2 = 0.001$ . Secondly, the value of the performance function varies with the kernel size as illustrated by the different



**Fig. 5.2.** Performance surface variation with kernel size (from [224]).

scales for the performance surface plots on the right. A changing kernel size during adaptation creates what we call a *dynamic performance surface*, that is, a performance surface that changes from iteration to iteration.

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 search algorithms that are affected include line searches and conjugate direction selection. In order to accommodate dynamic performance surfaces, the kernel size is kept constant at its current value during the execution of any sequence of search steps that require a performance comparison or a line search execution. Once these steps have been complete and the resulting set of weights is determined, kernel size adaptation is resumed. For some algorithms, this requires re-evaluation of the performance index so that it can be used in the next epoch as the basis for comparison.

Although this implies an increased computational burden for each epoch, the advantage of adapting the kernel size based on the new distribution of samples is fully realized.

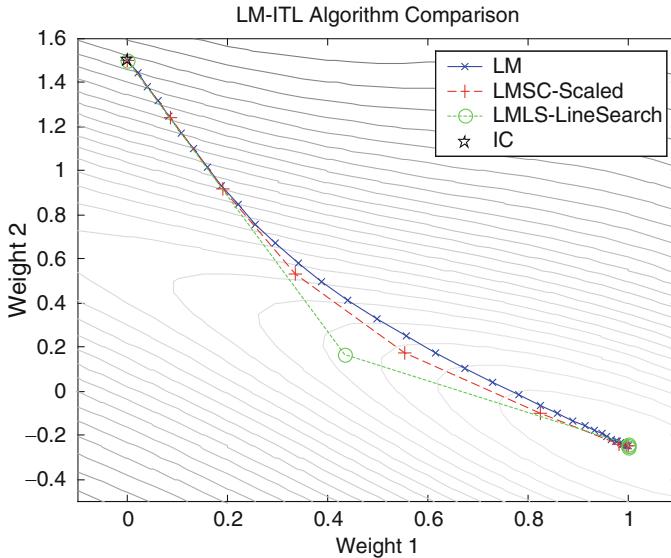
### Information Potential Versus Entropy as Performance Cost

The use of the MSE criterion produces 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 Eq. (3.8) allows the entropy to take on any real value. This characteristic can be difficult with some of the performance assessment indices described above; that is, Eq. (5.28) no longer works if both performance values are negative. Instead of trying to compensate for this fact (see [224]), we strongly suggest the use of the information potential  $V(E)$  (or  $V(\theta) - V(E)$  as in the MEE-SAS) as our cost function because they are always positive and so the only difference with respect to MSE is to remember that the goal of adaptation is to maximize  $V(E)$  (or minimize  $V(\theta) - V(E)$ ).

### Relative Error Versus Absolute Squared-Error

Although the MSE criterion provides an absolute error measure, ITL criteria provide only a relative measure of the error because the estimated entropy depends upon the kernel size and it is blind to the mean of the error distribution. The LM algorithm, designed specifically for the MSE criterion, uses the assumption of a quadratic performance surface along with the absolute MSE to determine both the direction and magnitude of weight change. This presents two problems. First, the computation of the MSE criterion is typically built into the LM algorithm implementations and requires a set of desired target outputs to compute the absolute MSE. For ITL training, a desired target set is not always available and even if it is available, it need not be the same dimensionality as the output space as we show in Chapter 6. These difficulties have been overcome by matching the mean of the model output to the desired mean and substituting the ITL information forces of Eq. (5.3) for the error term  $e$  in the gradient computation of the LM algorithm Eq. (5.26).

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 stepsize; however, in ITL the information forces only provide a relative measure of the error for each output sample. This mismatch in assumptions can lead to improper stepsizes during LM parameter adaptation as illustrated 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 described above. Although the algorithm seems



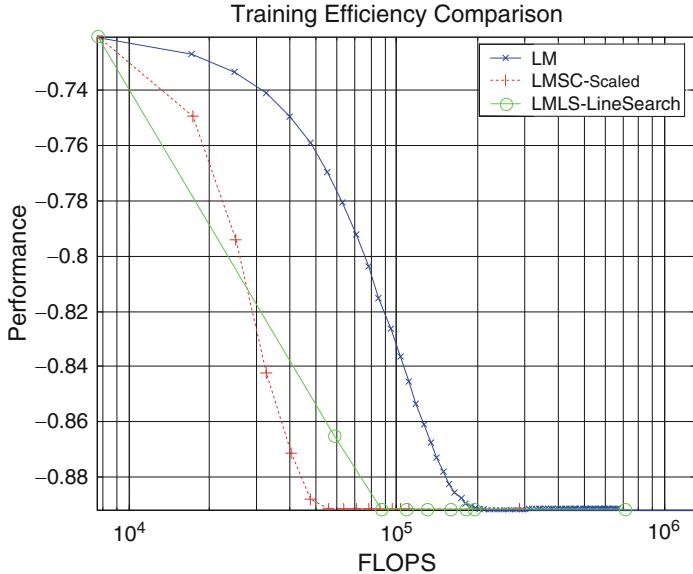
**Fig. 5.3.** Levenberg–Marquardt ITL algorithms: weight tracks from the same initialization condition (from [224]).

to be pointing in the appropriate direction at each step, many epochs are required for convergence compared to typical LM behavior due to an inappropriately small stepsize. To help alleviate this problem, two methods are proposed.

The first method, illustrated in the figure as “LM–SC”, simply scales the stepsize by scaling the ITL information forces in proportion to the number of samples,  $N$ . 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 empirically to be fairly robust. 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 (LM–LS) 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 ( $\alpha$  is determined using a line search)

$$\Delta \mathbf{w} = -\alpha \left[ \mathbf{J}^T \mathbf{J} + \mu \mathbf{I} \right]^{-1} \mathbf{J}^T \mathbf{e}. \quad (5.29)$$

The computational complexity of these three methods is compared in Figure 5.4. Notice how the proposed methods converge more efficiently than the basic LM method. This improved efficiency becomes more pronounced for larger values of  $N$ .



**Fig. 5.4.** Levenberg–Marquardt ITL algorithms: training efficiency (from [224]).

Due to the importance of second-order methods in adapting neural networks, we present in Table 5.1 the outline for parameter adaptation based on Levenberg–Marquardt with line search for ITL. In bold are the required changes when the cost is ITL.

### ITL Advanced Parameter Algorithm Summary

The sections above describe how to apply standard advanced parameter search techniques to ITL. For each problem encountered, a viable solution or workaround has been identified allowing ITL systems to be trained with more advanced algorithms. Table 5.2 summarizes the algorithms that have been applied to ITL along with the associated modifications that were required to arrive at a suitable implementation. Due to 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 floating-point operations (Flops). For the purpose of providing performance comparisons among the methods the frequency doubling problem is studied with a fixed kernel size using an ITL cost function (the  $QMI_{ED}$ ) discussed in the next chapter. However, we include it here because the goal of this comparison is the relative behavior of the search algorithms, not the cost function itself.

**Table 5.1.** LM–LS Flowchart (From [224])

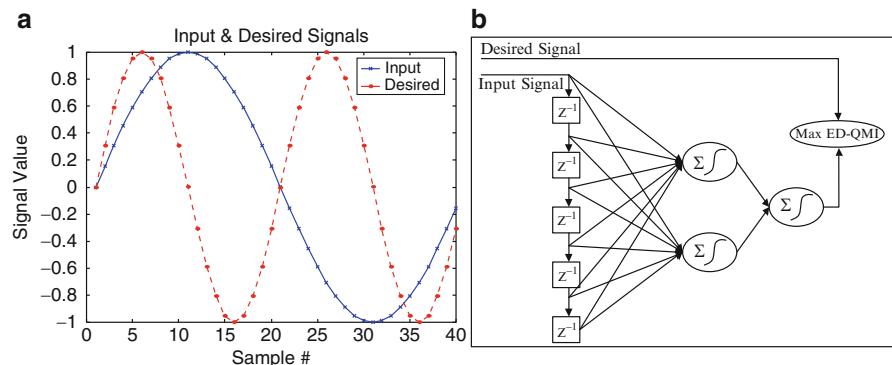
- 
1. Calculate Initial Network State:
    - a. **Compute Performance Using ITL Criterion (Instead of MSE).**
    - b. Compute Internal Network Signals (for Backpropagation).
    - c. **Substitute ITL-Based Information Forces for Error Terms.**
    - d. **Set the Kernel Size to Fixed Mode at Current 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  $\mu$  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 ( $\alpha$ ) 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 MSE).**
    - b. Compute Internal Network Signals (for Backpropagation).
    - c. **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  $\mu$ .
      - vi. Increment Epoch.
      - vii. Go to Step 2.
    - b. If Performance Declines:
      - i. Increase  $\mu$ .
      - ii. If  $\mu <= \mu_{\max}$  Go to Step 4.
      - iii. Else go to Step 3.
- 

### Case Study: The Frequency Doubler

This example is based on the frequency-doubling problem in [253]. 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 (i.e. a period of 20 samples). Five delay elements are used to create a six-input time delay neural network. 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. Figures 5.5a and b illustrate the input and desired signals and the network topology, respectively.

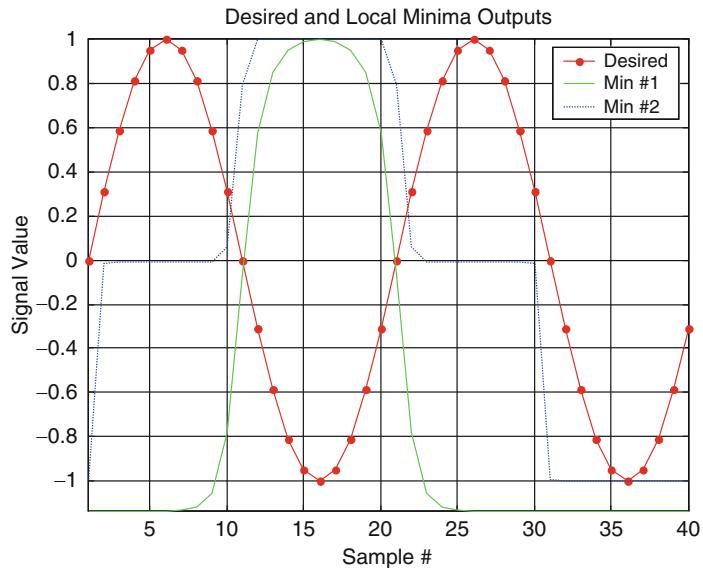
**Table 5.2.** ITL Advanced Parameter Search Algorithm Summary (From [224])

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

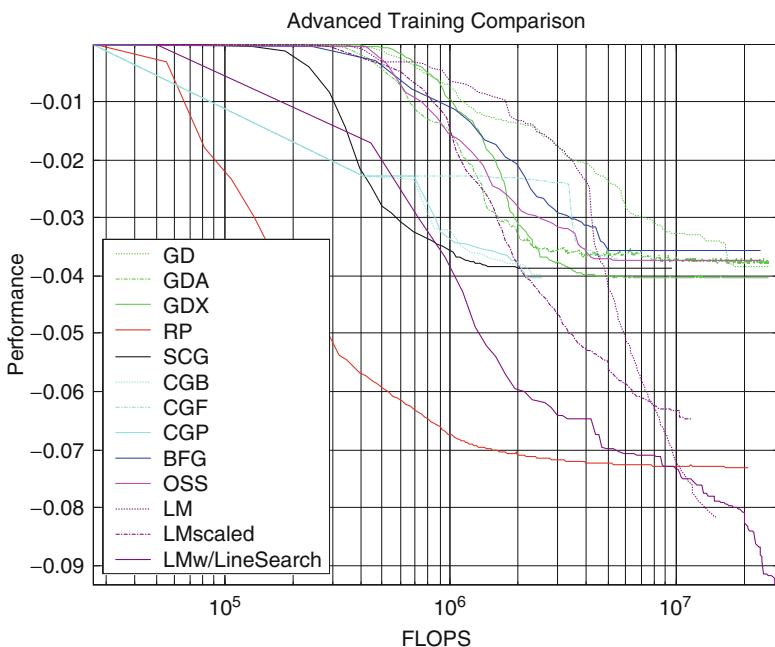
**Fig. 5.5.** Frequency doubler (a) input and desired signals; (b) TDNN (From [223]).

Training was conducted using a set of 30 different initial conditions for backpropagation. At least two local minima of the performance surface were identified during the training of this system, with a narrow global minimum. Due to the simplicity of the input one could identify the effect of the local minima, with one corresponding to a flat half-cycle (positive or negative depending upon the runs) and a staircase signal with level changes at each zero crossing of the desired signal as illustrated in Figure 5.6.

Figure 5.7 plots the average of the 30 learning curves against the number of floating point operations/s (FLOPS) required for the various search algorithms.



**Fig. 5.6.** Frequency doubler: local minima outputs (From [223]).



**Fig. 5.7.** Frequency doubler, parameter search comparison (from [223]).

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 illustrated by their high average terminal value of around  $-0.035$  to  $-0.04$ . After this group, the LM-SC, RP, LM, and LM-LS methods demonstrate progressively better terminal values. Of particular interest is the efficiency of the RP method, especially in the early stages of training. Although exhibiting a very good terminal average, RP has the steepest initial phase of learning. In terms of final performance, however, the LM-LS method performs the best overall.

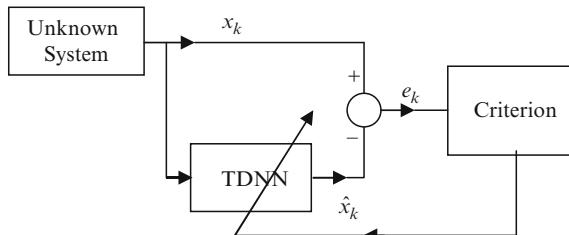
## 5.5 Application: Prediction of the Mackey–Glass Chaotic Time Series

The goal in dynamic modeling is to identify the nonlinear mapping that produced the given input–target data. This is traditionally achieved in a predictive framework as shown in Figure 5.8.

Minimization of MSE in the criterion block simply constrains the square difference between the original trajectory and the trajectory created by the adaptive system (TDNN in this case), which does not guarantee capturing all the structure about the underlying dynamics. Hence, we illustrate here the minimization of the EEC as a more robust criterion for dynamic modeling, and an alternative to MSE in other supervised learning applications using nonlinear systems, such as nonlinear system identification with neural networks [140].

Our first example is the single-step prediction of the well-known Mackey–Glass chaotic time series, which often serves as a benchmark in testing prediction algorithms in the literature. The Mackey–Glass system is a chaotic system (for  $\tau = 30$  or higher) given by

$$\dot{x}(t) = -0.1x(t) + \frac{0.2x(t-30)}{1+x(t-30)^{10}} \quad (5.30)$$

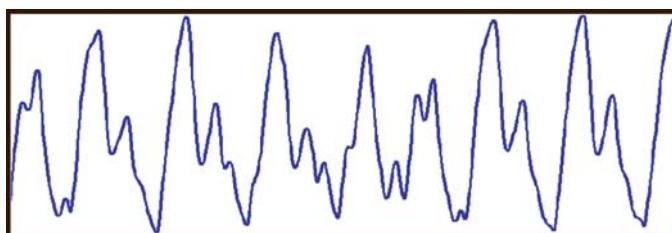


**Fig. 5.8.** Time-delay neural network prediction scheme.

that was proposed to model irregular behavior in biological systems [113]. After transients die out the system response approaches its limit dynamics (the attractor). The dimension of the attractor for a chaotic system is normally a real number, and it is possible to embed the attractor onto a sufficiently large Euclidean space such that there is a one-to-one mapping (with a unique inverse) between the trajectories in the attractor and the system output (i.e. the attractor trajectories never cross each other). If the dimension of the attractor is  $D$ , the embedding space dimension is selected higher than  $2D + 1$  [140]. When the embedding is implemented with time delays, the number of time delays should be  $2D$ , which for the MG30 is 7 [253]. For our simulations, we use samples drawn at  $T = 1$  s intervals from the MG30 system of Eq. (5.30), using the Runge–Kutta method with time-step equal to 0.1 s, and then the generated series was down-sampled by 10, to get the desired sampling period of 1 s. Figure 5.9 illustrates a segment of the MG30 time series.

In all the following simulations regarding the MG30 data, we used 200 samples for training and 10,000 new test samples are generated using a different initial condition, thus are from a different trajectory on the same attractor.

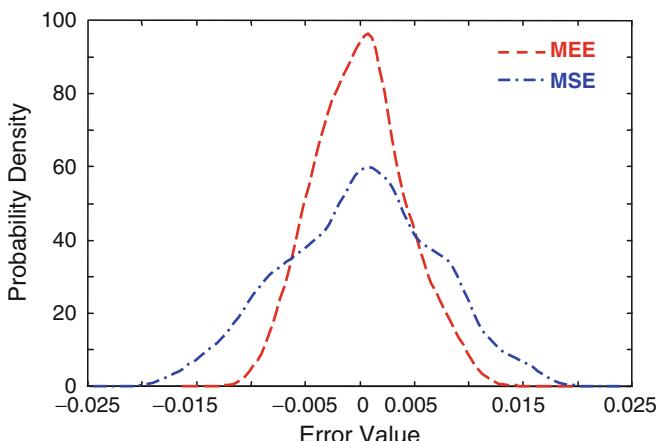
As the aim of our first set of simulations is to compare the generalization properties of learning with MSE versus learning with MEE, we train two identical TDNNs on the same data: one of them uses MSE as the criterion and the other uses EEC. In addition, in order to make sure that the results we obtain are not dependent on the specific TDNN architecture we choose its capabilities; we include eight different two-layer TDNNs in each group with seven delays and whose number of hidden neurons varies from three to ten. To increase the speed of training for all 16 TDNNs we use the scaled conjugate gradient approach explained above. However, in order to avoid local minima we take the Monte Carlo approach to select the initial conditions for the weight vectors and use 1000 (uniformly distributed) randomly selected sets of weights for each TDNN. After all 16 TDNNs are trained, the weight vectors of the solution that yielded the smallest MSE were selected to be the optimal solution. Similarly, the optimal weight vector for the TDNNs trained with the EEC criterion was selected as the run that yielded the smallest error entropy.



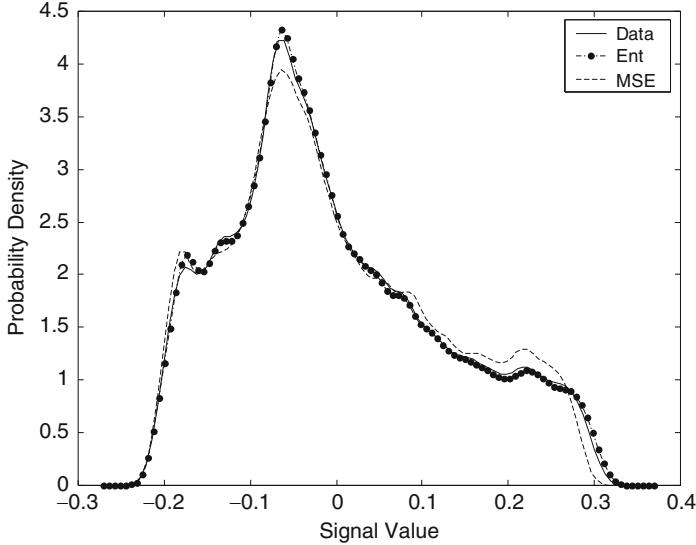
**Fig. 5.9.** A segment of the Mackey-Glass ( $\tau = 30$ ) system.

Afterwards, these solutions were iterated a couple of more epochs to guarantee their convergence to the minimum; in fact, visual inspection of the learning curves for all TDNNs showed that with the conjugate gradient approach, all TDNNs using the BP algorithm converged in less than 100 iterations and all TDNNs using the MEE-BP algorithm criterion converged in less than 30 iterations. It must be noted, however, that the computational complexity of the MEE-BP is greater than that of the BP. The Gaussian kernel was used to estimate entropy in all simulations with size  $\sigma = 0.01$  because it provided experimentally the best results. In addition, the output bias of the linear output PE is set as the sample mean of the desired output for both MSE and EEC criteria. In this first set of simulations, we use Renyi's quadratic entropy definition. Surprisingly, the MEE-BP algorithm achieves smaller variance for all sizes of the network except for six hidden neurons. For this reason, in the following, we elaborate on this special case, where the TDNN has six hidden PEs.

The first question is how to compare the two approaches fairly. Indeed if we use MSE as our comparison criterion, we may be favoring the TDNN trained with MSE, and vice versa. A possible approach used here evaluates how close to the delta function are the error PDFs trained with the two distinct methods. One can expect that the delta function will not be achieved due to imprecision in the estimation of optimal parameters, and because the mapping capabilities of the TDNN with a finite number of units may not include the system that generated the MG30 system. Figure 5.10 shows the estimated error PDFs of the best-performing TDNNs for each cost. Clearly, the TDNN that is trained using the MEE-BP algorithm provided errors more concentrated around zero (i.e. higher number of smaller errors and fewer large errors), which corresponds to a better model. In this case there are no noticeable ripples in the error PDF.



**Fig. 5.10.** Comparisons of reconstructed error PDFs for MEE and MSE training (from [87]).



**Fig. 5.11.** Probability density estimates of the 10000-sample MG30 test series (solid) and its predictions by MEE-trained (thick dots) and MSE-trained (dotted) TDNNs. All PDFs are normalized to zero-mean (from [87]).

Figure 5.11 shows the estimated PDF of the data generated in the test set by the two different models and clearly demonstrates that the PDF of the TDNN trained predictions with MEE-BP follows very closely the PDF of the data. In this case it seems an  $L_1$  fit, but this is not guaranteed, because the EEC criterion favors a large concentration of errors (delta functions have negligible entropy) so there are many possible local minima, and how to avoid them without using brute force is still unknown.

Our second set of simulations was aimed at investigating the effect of entropy order on the performance of the final solution obtained. The effect of the kernel size was studied as well. For each set of free parameters (kernel size and entropy order) we ran 100 Monte Carlo runs using randomly selected initial weight vectors. At the end of the training, which used 200 samples, the information potential of the error on the test set consisting of 10,000 samples corresponding to each TDNN was evaluated using a Gaussian kernel of size  $\sigma = 10^{-3}$  to provide a good basis for a comparison (i.e. there are plenty of data to obtain a good estimate of the true statistical quantity).

For the final error signals obtained, this value of the kernel size allows the kernels to cover on average 10 samples in  $3\sigma$  (this is another simple rule of thumb to establish the kernel size). The results are summarized in Table 5.3 in the form of normalized information potentials (maximum value is one). There are a total of 12 trained TDNNs, using the designated entropy orders and kernel sizes given in the first column. The performances of these TDNNs are then evaluated and compared using two different entropy orders, presented in

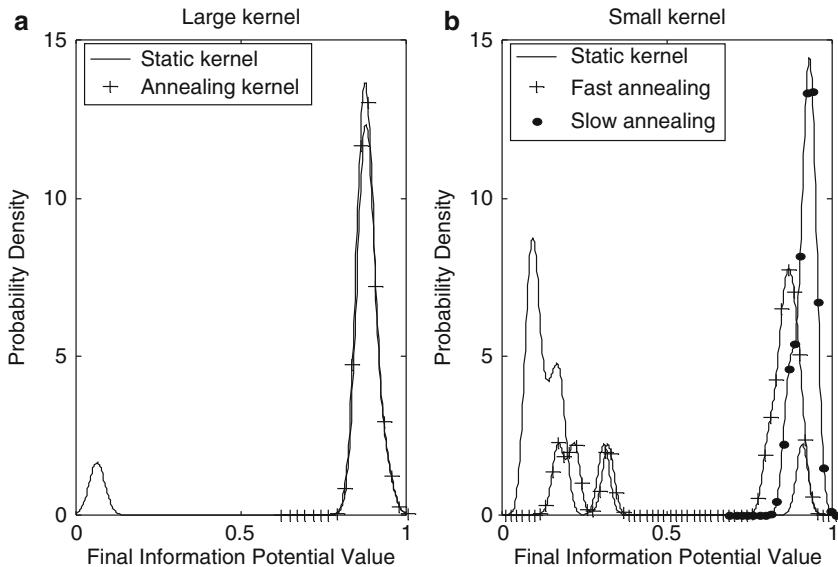
**Table 5.3.** Normalized Information Potential Evaluation (From [87])

Training Parameters	Evaluation Parameters	$V_\alpha(e)$	$V_\alpha(e)$
		$\alpha = 1.01$	$\alpha = 2$
		$\sigma = 10^{-3}$	$\sigma = 10^{-3}$
$\alpha = 1.01$	$\sigma = 0.01$	0.976	0.099
	$\sigma = 0.1$	0.976	0.104
	$\sigma = 1$	0.969	0.047
$\alpha = 1.5$	$\sigma = 0.01$	0.977	0.112
	$\sigma = 0.1$	0.977	0.109
	$\sigma = 1$	0.976	0.105
$\alpha = 2$	$\sigma = 0.01$	<b>0.979</b>	<b>0.135</b>
	$\sigma = 0.1$	<b>0.979</b>	<b>0.133</b>
	$\sigma = 1$	<b>0.978</b>	<b>0.126</b>
$\alpha = 3$	$\sigma = 0.01$	0.977	0.124
	$\sigma = 0.1$	0.977	0.117
	$\sigma = 1$	0.976	0.105

each column. The first thing to notice is that, as expected, the information potential estimates (and therefore the entropy estimates) change a lot with the selected  $\alpha$  and kernel size, the two free parameters of the method. This reflects the fact that the estimates are a function of the kernel size, and therefore quoting an estimated entropy value at the optimal solution should be avoided. But notice that once the parameters are selected and during the adaptation process, this dependence is immaterial, because the goal of adaptation is to seek the extremes (either maximum or minimum) for the same data with a preselected choice of the free parameters.

It is comforting to see that irrespective of the column (entropy order in the evaluation) the best results occur for the networks trained with the same entropy order ( $\alpha = 2$  in this case), so this means that the IP estimator is accurate when the number of samples is large. But the difference in values across the rows means that different optimal points in parameter space are obtained for different  $\alpha$ , and the best for this example is  $\alpha = 2$ . The variability of the IP estimate within each  $\alpha$  for the range of kernel sizes tried is small, perhaps showing that using smaller kernel sizes in training improves the final set of optimal coefficients. The dynamic range of the IP values is much higher for  $\alpha = 2$  than  $\alpha = 1.01$ .

Our third set of simulations investigates the validity of the conjecture on the global optimization capabilities of the MEE algorithm. In these simulations, we use the quadratic entropy criterion on the MG30 training data again. This time, however, the size of the Gaussian kernel is annealed during the training from a large value to a smaller one. Once again the Monte Carlo approach is taken with 100 randomly selected initial weight vector assignments.



**Fig. 5.12.** Probability distributions of the final normalized information potential values obtained on the training set when the kernel size is (a) large; static kernel (solid), slow annealing (+); (b) small; static kernel (solid), fast annealing (+), slow annealing (dots) (from [88]).

The results of these experiments are summarized in Figure 5.12 as a PDF estimate of the final normalized information potential values (so that the maximum value is one) obtained on the training data. In Figure 5.12a, the distributions of the final performances for two experiments (fixed and annealed kernel sizes) are shown. In the static kernel case the kernel size is kept fixed at  $\sigma = 10^{-2}$ , whereas the changing kernel has an exponentially annealed size  $\sigma = 10^{-1} \rightarrow 10^{-2}$ , during a training phase of 200 iterations. For the large static kernel size of  $\sigma = 10^{-2}$ , approximately 10% of the time the algorithm got trapped in a local maximum of the information potential with a normalized value of about 0.1. The annealed kernel size algorithm avoided this local optimum in all the runs and achieved a better (possibly the global) maximum, with a normalized value of 0.9.

In Figure 5.12b, the distributions of the performances for three experiments are shown, but now the static kernel has a size of  $\sigma = 10^{-3}$  throughout the training. The slow- and fast-annealed kernels, on the other hand, have exponentially decreasing sizes of  $\sigma = 10^{-1} \rightarrow 10^{-3}$  for a training phase of 500 and 200 iterations, respectively. This annealing scheme is the same for all initial conditions. Because the kernel size is smaller now, we can expect more local maxima in the normalized information potential surface, but more accurate performance if global maximum is achieved.

In this small kernel case with  $\sigma = 10^{-3}$ , it is observed that the static kernel gets trapped in local maxima quite often (90% of the time), whereas the fast

annealed kernel shows some improvement in avoiding local optima (70% of the time achieves global optimum), and eventually the slow annealed kernel consistently achieves the global maximum (100% of the time).

These experiments showed that, by annealing the kernel size, one is likely to improve the algorithm's chances of avoiding local optimum solutions. However, there is no prescription yet for how to anneal the kernel size. The exponential annealing scheme and the decay rates assumed in the above simulations were determined by trial and error.

## 5.6 Application: Nonlinear Channel Equalization

In digital communications, the transmission of messages is plagued by the intersymbol interference (ISI) due to the finite bandwidth of the channel. The most popular equalizer is the linear transversal equalizer (LTE), trained to minimize the MSE between its output and the desired sequence by means of the LMS or the recursive least square algorithm [255]. An interesting and powerful alternative to the LTE is the decision feedback equalizer (DFE). In this case, the past decisions are included in the equalization process to improve the margin against noise and performance, mainly in channels with deep nulls. Although the DFE structure is nonlinear, it can only cope with very moderate nonlinear distortion. Moreover, it suffers from error propagation due to the feedback part. The received signal  $x_i$  at the input of the equalizer can be expressed as

$$x_i = \sum_{k=0}^{n_c} h_k s_{i-k} + e_i, \quad (5.31)$$

where the transmitted symbol sequence is assumed to be an equiprobable binary sequence  $\{1, -1\}$ ,  $h_k$  are the channel coefficients (we assume here an FIR channel), and the measurement noise  $e_i$  can be modeled as zero-mean Gaussian with variance  $\sigma^2_n$ . The equalization problem reduces to correctly classify the transmitted symbols  $s_i$  based on the observation vector. For instance, LTE estimates the value of a transmitted symbol as

$$\hat{s}_{i-d} = \text{sgn}(y_i) = \text{sgn}(\mathbf{w}^T \mathbf{x}_i), \quad (5.32)$$

where  $y_i = \mathbf{w}^T \mathbf{x}_i$  is the output of the equalizer,  $\mathbf{w} = [w_0, \dots, w_{M-1}]^T$  are the equalizer coefficients,  $\mathbf{x}_i = [x_i, \dots, x_{i-M+1}]^T$  are the observations and  $d$  is the equalizer delay. The LTE implements a linear decision boundary; however, it is well known that even if the channel is linear, the optimal (Bayesian) decision border is nonlinear, which becomes more noticeable when the noise increases.

On the other hand, when the channel is nonlinear, in order to eliminate the ISI, it is necessary to consider a nonlinear equalizer. Recently, artificial neural networks have been proven to be attractive alternatives for nonlinear equalization, in particular, the multilayer perceptron (MLP) and the radial

basis function (RBF) have demonstrated good performance [31]. In this case, the output of the equalizer is given by

$$y_i = g(\mathbf{x}_i, \mathbf{W}), \quad (5.33)$$

where  $g(\cdot)$  is a nonlinear mapping and  $\mathbf{W}$  denotes the parameters of the equalizer. After the mapping, a hard threshold is still needed in order to decide the symbols; in this way, Eq. (5.32) can be viewed as a mapping from the input space to an output space, where the classification becomes possible and hopefully easier. Here an MLP will be employed to perform that mapping. Assuming a single hidden layer MLP, Eq. (5.33) reduces to

$$y_i = \mathbf{w}_2^T \tanh(\mathbf{W}_1 \mathbf{x}_i + \mathbf{b}_1) + b_2, \quad (5.34)$$

where  $\mathbf{W}_1$  is an  $N \times M$  matrix connecting the input layer with the hidden layer,  $\mathbf{b}_1$  is an  $N \times L$  vector of biases for the hidden layer PEs,  $\mathbf{w}_2$  is an  $N \times L$  vector of weights connecting the hidden layer to the ouptut PE and  $b_2$  is the output bias. The training of this structure to minimize the MSE criterion can be done using the BP algorithm [253].

Alternatively, Renyi's quadratic entropy can also be used as the cost function. Obviously, the minimum of Renyi's entropy is obtained when  $p(e) = \delta(e - \varepsilon)$  for any  $\varepsilon$  (i.e., when the error is a constant signal). On the other hand, the PDF of the equalizer's output  $y = s + e$ , given the training sequence  $s$  (considered as deterministic), is

$$p(y|s) = p_e(y - s). \quad (5.35)$$

In this way, the output of the equalizer converges to  $y = s + \varepsilon$  with probability one. In practice, a finite training sequence is used; in this case, by maximizing the information potential, each error sample interacts with all other errors, pushing the solution towards a constant error signal. The constant term has no influence because it can be easily eliminated using an additional bias term in the equalizer. This argument supports the use of an error entropy minimization criterion in equalization problems, but a closer analysis shows an even better picture.

To gain some more insight into the appeal of the error entropy criterion for this problem, it is interesting to write the information potential as a function of the output of the equalizer (linear or nonlinear)  $y_i$ . Considering a binary signal, the set of outputs  $y_i$  for the training set can be partitioned according to the desired output  $s_{i-d} = \pm 1$  into the following two subsets:

$$R^{(\pm 1)} = \{y_i, s_{i-d} = \pm 1\}. \quad (5.36)$$

Now, taking into account that

$$e_i - e_j = s_{d-i} - s_{d-j} + y_j - y_i. \quad (5.37)$$

it is easy to show that

$$\begin{aligned}\hat{V}(y) = & \sum_{i,j} \sum_{\in R^{(+1)}} G(y_i - y_j) + \sum_{i,j} \sum_{\in R^{(-1)}} G(y_i - y_j) \\ & + 2 \sum_{i \in R^{(+1)}} \sum_{j \in R^{(-1)}} G(y_i - y_j - 2).\end{aligned}\quad (5.38)$$

The first two terms in Eq. (5.38) are maximized when  $y_i = y_j$  for  $i, j \in R^{(+1)}$  and  $i, j \in R^{(-1)}$  respectively. This process can be viewed as minimizing the “intraclass” output entropy; that is, the equalizer tries to cluster the outputs in delta functions for inputs belonging to  $R^{(+1)}$  and  $R^{(-1)}$ . On the other hand, the third term is maximized when  $y_i - y_j = 2$ , for  $i \in R^{(+1)}$  and  $j \in R^{(-1)}$ ; therefore, it tries to separate the outputs for each class. As a comparison, the MSE criterion in terms of the equalizer outputs is given by

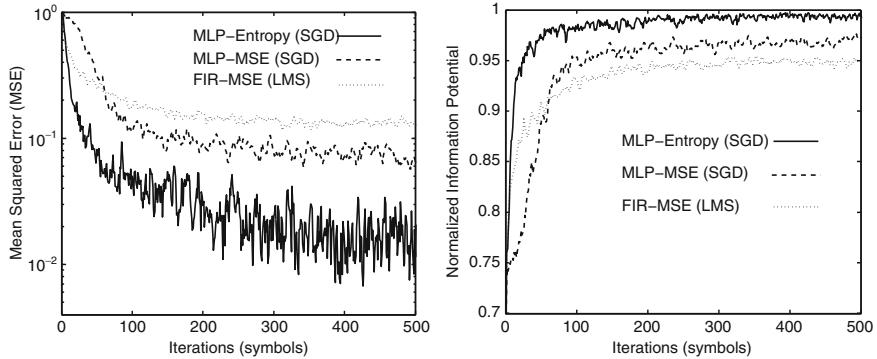
$$MSE(y) = \sum_{i \in R^{(+1)}} (1 - y_i)^2 + \sum_{i \in R^{(-1)}} (1 + y_i)^2.\quad (5.39)$$

It can be concluded that the entropy criterion forces additional constraints by exploiting the relationship between each pair of equalizer outputs. Moreover, although the MSE criterion forces a constant modulus for the output signal as in Eq. (5.39), the entropy criterion ensures that the difference between the outputs for the two classes has a constant value. The advantages of MEE for equalization are particularly useful for the nonlinear channel case [31].

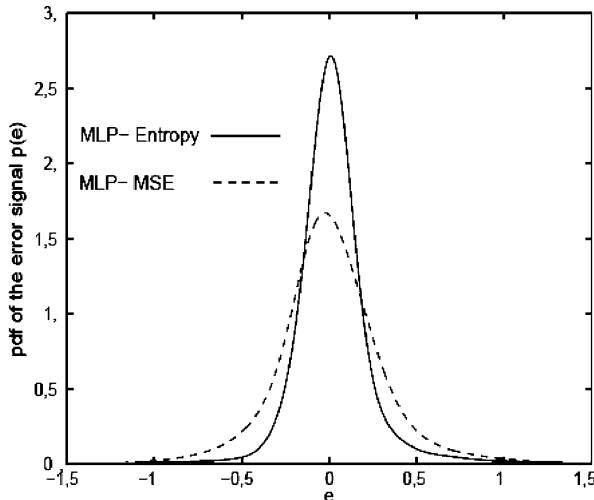
The nonlinear channel used here is composed of a linear channel followed by a memoryless nonlinearity often encountered in digital satellite communications [279] and as nonlinear channel models for digital magnetic recording [280]. The linear channel considered is  $H(z) = 0.3482 + 0.8704z^{-1} + 0.348z^{-2}$ , and the nonlinear function applied is  $z = x + 0.2x^2 - 0.1x^3$ , where  $x$  is the linear channel output. Finally, white Gaussian noise for SNR 16 dB was added. The nonlinear equalizer structure is an MLP with seven PEs in the input layer and three PEs in the hidden layer, and the equalization delay is  $d = 4$ .

For this example, the online MEE-BP algorithm is applied with a short sliding window of just  $N = 5$  error samples. At each iteration a single step was taken. For both criteria, a fixed stepsize  $\eta = 0.01$  was used, which is the largest stepsize for which the algorithms converged in all trials. The results provided by a linear (FIR) equalizer with coefficients and trained with an MSE criterion were also obtained. In this case, a conventional LMS algorithm with a fixed stepsize was used.

Figure 5.13 shows the convergence of the normalized information potential and the MSE evaluated over the sliding window for the three algorithms. These results were obtained by averaging 100 independent simulations. It can be seen that the MLP trained with the MEE-BP achieves the best results, and it also provides the fastest convergence, whereas the linear equalizer is not able to



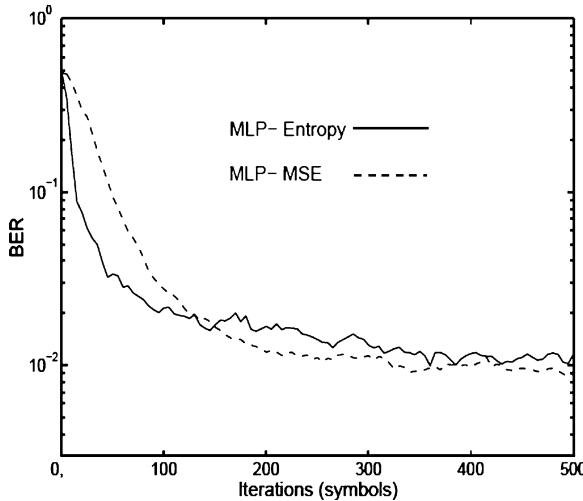
**Fig. 5.13.** Convergence of the different equalizers in terms of MSE and IP (from [281]).



**Fig. 5.14.** Error PDF for the MLP equalizer trained with MSE and MEE (from [281]).

remove the nonlinear ISI part. It is interesting that even though the entropy criterion does not directly minimize the MSE it achieves a lower MSE than direct minimization of this criterion.

The explanation is that, in comparison to the MSE, the entropy criterion yields a spiky error with more abrupt changes (higher kurtosis) but with a lower MSE, and this was experimentally verified, see Figure 5.14.



**Fig. 5.15.** Convergence of the bit error rate through iterations for MSE and MEE (from [281]).

The error sequence PDF was estimated using the Parzen windowing method with  $\sigma^2 = 0.01$  and as expected the minimization of the error entropy tries to push the PDF of the error closer to a delta function.

The important parameter in communications is not the equalization but the bit error rate (BER). The convergence of the BER with the number of training symbols is shown in Figure 5.15. Experience with the MSE criterion shows that better equalization implies a better (smaller) BER. However, the entropy criterion achieves a very fast decrease in BER, but the final BER is slightly worse than the BER obtained by the MSE criterion. This result was not expected, given the better equalization of the channel achieved with the EEC, and it may be explained by the spikiness of the error achieved with minimization of error entropy [281]. This problem requires a better control of the kernel size. Our suggestion at this point is to switch between MEE and MSE during equalization but an entropy order  $\alpha < 1$  has the potential to improve the BER, although it can slow down the training. This example demonstrates that each practical application has its own peculiarities that require attention when testing a new cost criterion.

## 5.7 Error Correntropy Criterion (ECC) in Regression

Recall that Eq. (3.36) defines still another cost function (the ECC) that is based on the correntropy criterion. In Section 3.6, ECC was shown equivalent to the EEC if one works with the difference in errors instead of the individual errors. The appeal of the ECC and its MCC algorithm is its much faster

computation when compared with EEC, the automatic centering of the error PDF at zero, but one loses the information-theoretic interpretation. Just by looking at the CIM metric in Figure 3.9 we can anticipate that the ECC may pose problems during adaptation because the errors saturate and so the search algorithms based on this cost function become less sensitivity to change in errors (which gives rise to plateaus also called flat spots, or even local minima). To elucidate the practical application of these two alternate costs, we compare them on a simple linear and nonlinear regression problem, along with the more conventional LAR (least angle regression) and the bisquare weights (BW) mentioned in Chapter 3. The same algorithms can be used for filtering, although we do not pursue this application here.

## Linear Regression

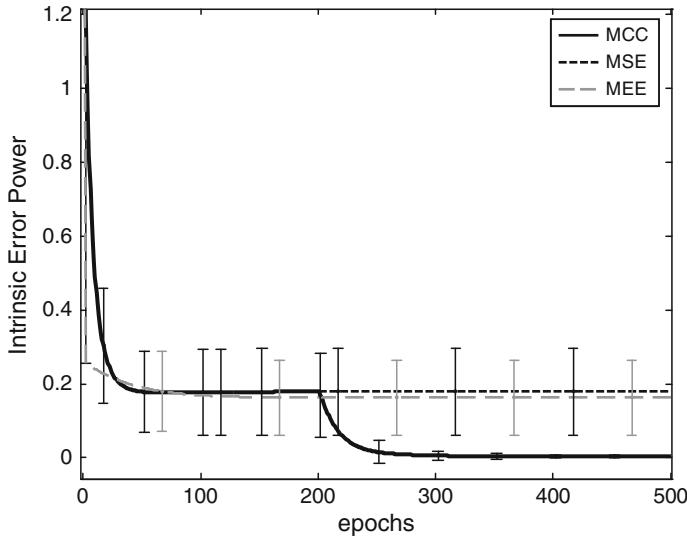
Let us consider the general model of regression  $Z = f(X) + v$  where  $f$  is an unknown function,  $v$  is a noise process and  $Z$  is the observation. Here  $\{(x_i, z_i)\}_{i=1}^N$  are the training data. A parametric approximator  $g(x; w)$  (specified below) is used to discover this function and alleviate the effect of noise as much as possible. Let the noise probability density function be an impulsive Gaussian mixture (Middleton model)  $p_\nu(\nu) = 0.9G(0, 0.1) + 0.1G(4, 0.1)$ . In MSE, the optimal solution is found by Eq. (3.2) and for the MCC algorithm, the optimal solution is found as

$$\max J(w) = \frac{1}{M} \sum_{i=1}^N k_\sigma(g(x_i; w) - z_i) \quad (5.40)$$

The first example implements a first-degree polynomial system for simplicity; that is,  $g(x; w) = w_1x + w_2$ .  $f(x) = ax + b$  with  $a = 1$  and  $b = 0$ . Inasmuch as the ultimate performance of the MCC algorithm is under investigation here, the kernel size is chosen by systematically searching for the best result. Performance sensitivity with respect to kernel size is quantified experimentally in Table 5.4 and is compared with the kernel size estimated by Silverman's rule. The data length is set small on purpose:  $N = 100$ . Steepest descent is used for

**Table 5.4.** Regression Results Summary (From [201])

Algorithms	$a$	Standard Deviation of $a$	$b$	Standard Deviation of $b$	Intrinsic Error Power	Standard Deviation of Intrinsic Error Power
MSE	1.0048	0.1941	0.3969	0.1221	0.1874	0.1121
MCC	0.9998	0.0550	0.0012	0.0355	0.0025	0.0026
MEE	0.9964	0.0546	0.3966	0.1215	0.1738	0.1049
LAR	1.0032	0.0861	0.0472	0.0503	0.0072	0.0066
BW	1.0007	0.0569	0.0010	0.0359	0.0025	0.0025

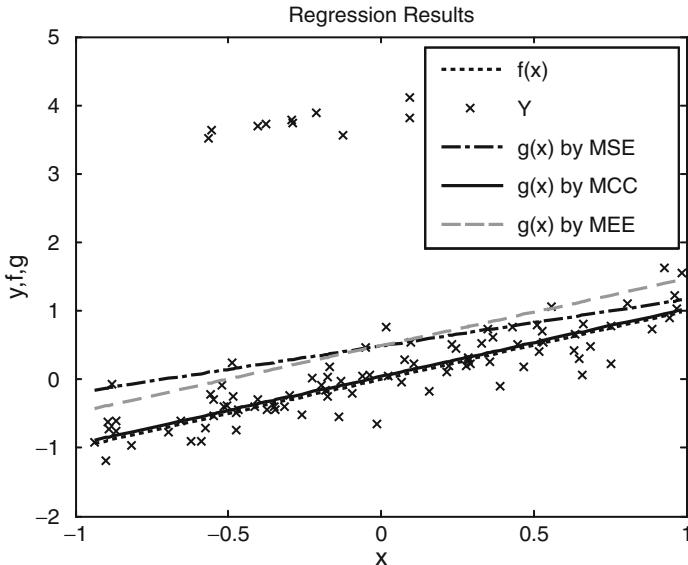


**Fig. 5.16.** Average learning curves with error bars of MCC, MSE and MEE (from [201]).

both criteria. Under the MSE criterion, the learning rate is set to 0.001 and the system is trained for 500 epochs (long enough to guarantee it reaches its global solution). For the MCC algorithm, we first train the system with the MSE criterion during the first 200 epochs (which is an extreme case of a large kernel size in kernel size annealing), and switch the criterion to ECC during the next 300 epochs. The learning rate is set to 0.001 and the kernel size is 0.5 which performs best on test data. We run 50 Monte Carlo simulations for the same data by changing the initial conditions. The average estimated coefficients for MSE are [0.484 0.679] and [0.020 0.983] for MCC. The average learning curves for the intrinsic error power ( $E[(g(X; w) - f(X))^2]$ ) are shown in Figure 5.16, along with its standard deviation. We observe that after the switching between criteria, the learning curve drops rapidly to a new value with small variability after 300 iterations, which means that the ECC criterion found a better set of weights.

When the MSE criterion is used,  $g(x)$  is shifted by the nonzero-mean noise and slanted by the outliers due to the global property of MSE (Figure 5.17). Now we see the importance of correntropy with its local property. In other words, correntropy has the ability of being insensitive to the shape of the noise PDF tail. For comparison, we also include the result of the MEE algorithm with the bias set at the mean of the desired response. Although MEE is also insensitive to outliers, the error is not symmetric and the resulting regressor will be biased, which explains the large final error.

Although the main purpose of this example is to highlight the robustness of correntropy, we also compare performance with the existing ro-



**Fig. 5.17.** Regression results with criteria of MSE, MCC, and MEE respectively. The observation  $Y$  is corrupted with positive impulsive noise; the fit from MSE (dash-dot line) is shown shifted and skewed; the fit from MCC (solid line) matches the desired (dotted) quite well; the fit from MEE is shifted but not skewed (from [201]).

bust fitting methods such as least absolute residuals (LAR) (which uses an  $L_1$  norm penalty) and bi-square weights (BW) [128]. The parameters of these algorithms are the recommended settings in MATLAB. All the results (50 Monte Carlo for each) are summarized in Table 5.4 in terms of intrinsic error power on the test set. Recall that the intrinsic error power compares the difference between the model output and the true system (without the noise). The performance of MCC is much better than LAR, and when regarded as a  $l_1$  norm alternative, correntropy is differentiable everywhere and to every order. Furthermore, notice that there is no threshold for MCC, just the selection of the kernel size. Moreover, the algorithm complexity of MEE is  $O(N^2)$  whereas MSE, MCC, BS and LAR are all  $O(N)$ .

We have to remember that the kernel size plays an important and perhaps contradictory role here, representing the statistics of the data and attenuating the outlier noise. In [201] we test the performance as a function of the kernel size, mean and variance of outliers, and the major conclusion is that the MCC algorithm performs at the same level or better than BS and LAR and that there is a relatively large range of kernel sizes that provides very similar performance. Moreover, the standard Silverman's rule for density estimation falls within the good performance range. If the data are plentiful, there is basically no compromise and good outlier rejection can be achieved, however,

a compromise must be struck for small datasets (because the kernel size cannot be made arbitrarily small). Nevertheless, MCC using large kernel sizes will perform no worse than MSE due to the correntropy unique metric structure. The shortcoming of the MCC algorithm is in the adaptation, because of its local nature. We recommend that the kernel size be started large compared with Silverman's rule for the data, and slowly annealed to the Silverman value. Otherwise the adaptation is very slow, or can even stall due to local minima. Alternatively, one can start the adaptation with MSE to bring the weights to a reasonable value close to the optimal solution, and then switch to the ECC cost as we illustrated in the example.

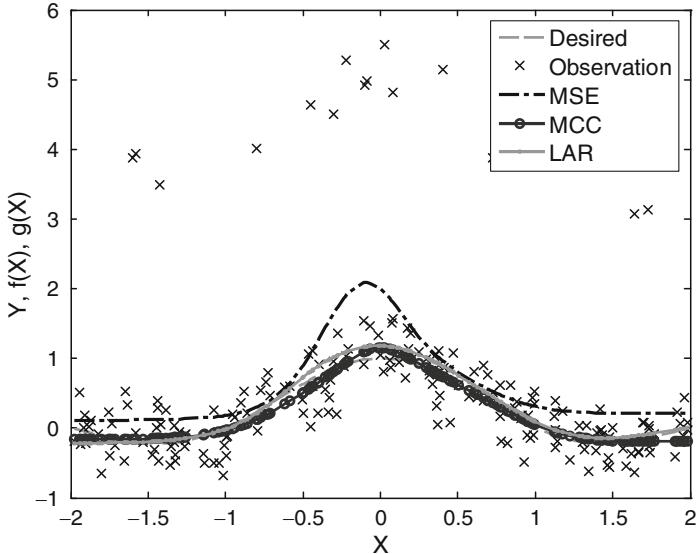
## Nonlinear Regression

A second, more complex and nonlinear, regression experiment is conducted to demonstrate the efficiency of the MCC algorithm. Let the noise PDF be the same as above and  $f(X) = \text{sinc}(X)$ ,  $X \in [-2, 2]$ . An MLP is used as the function approximator  $g(x; w)$  with one input unit, seven hidden units with  $\tanh$  nonlinearity and one linear output. The MLP is trained with MEE-BP (Eq. (3.36)), using an online update implemented with the stochastic gradient (see Eq. (4.61)). The data length is  $N = 200$ . Under the MSE criterion, the MLP is trained for 500 epochs with learning rate 0.01 and momentum rate 0.5. Under the LAR criterion, 600 epochs are used with learning rate 0.002 and momentum rate 0.5. In the MCC case, the MSE criterion is used for the first 200 epochs and switched to MCC for the next 400 epochs with learning rate 0.05 and momentum rate 0.5. Different values of  $\varepsilon$  are tried to test the efficiency of MCC against LAR. Fifty Monte Carlo simulations are run for each value. The results are shown in Table 5.5. The kernel size in MCC is chosen as  $\sigma = 1$  for best results. A nice feature of MCC is that it can attain the same efficiency as MSE when the noise is purely Gaussian due to its unique property of "mix norm" whereas LAR can not.

Experimentally, the behavior of MCC for nonlinear systems is very similar to the linear case considered above. Figure 5.18 shows the good fit to the

**Table 5.5.** Nonlinear Regression Results Summary (From [201])

$\varepsilon$	Intrinsic Error by MCC	Intrinsic Error by MSE	Intrinsic Error by LAR	Standard Deviation of Intrinsic Error Power by MCC	Standard Deviation of Intrinsic Error Power by MSE	Standard Deviation of Intrinsic Error Power by LAR
0.1	0.0059	0.2283	0.0115	0.0026	0.0832	0.0068
0.05	0.0046	0.0641	0.0083	0.0021	0.0325	0.0041
0.01	0.0039	0.0128	0.0058	0.0017	0.0124	0.0025
0	0.0040	0.0042	0.0061	0.0019	0.0020	0.0028



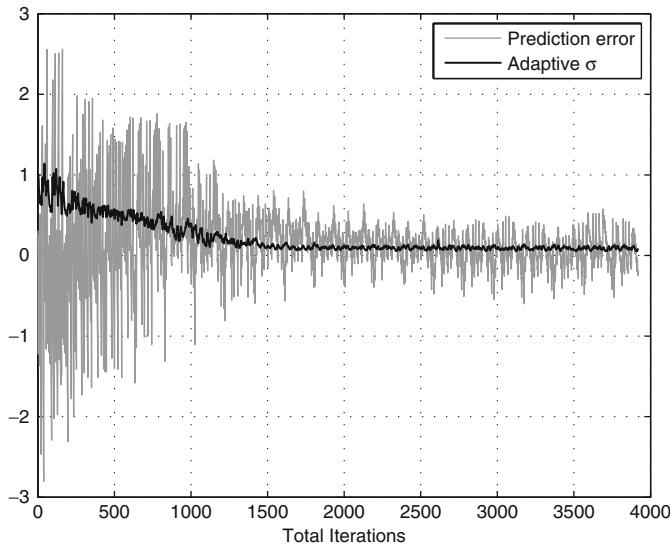
**Fig. 5.18.** Nonlinear regression results by MSE, MCC and LAR respectively (from [201]).

true regression of MCC and LAR across the whole domain, and the poorer approximation obtained with MSE due to the outliers.

## 5.8 Adaptive Kernel Size in System Identification and Tracking

In Chapter 3 we introduced the concept of online adaptation of the kernel size, and here we illustrate its performance in nonlinear adaptive filtering. For our first application, we train a TDNN with the MEE-BP algorithm (see Section 5.2) with an adaptive kernel, for prediction the Lorenz system output which is a well-known chaotic dynamical system [204]. We focus on evaluating the effects of using different values of  $\sigma$ , including the adaptive kernel size algorithm of Section 3.7. The kernel size update technique is incorporated by continuously updating  $\sigma$  using Eq. (3.70) and the weights, but using a smaller (10 times) learning rate for  $\sigma$ . Using this framework, we trained a 6-tap TDNN, with a single hidden layer and six processing elements in the hidden layer with a sigmoid nonlinearity. Two hundred time samples from the Lorenz attractor were used in the one-step-ahead prediction training set. Therefore, the neural network was trained to predict the next sample in the time series, by using the six previous samples. The system was trained for 20 epochs over the training set.

Figure 5.19 shows the evolution of the prediction error (difference between the desired sample and the predicted value) over all the iterations (cumulated



**Fig. 5.19.** Evolution of prediction error during the training of the neural network. The kernel width adapts to the values of the error.

over the 20 epochs). As expected, the kernel widths adapt to the values of the error. The adaptation automatically results in annealing of the kernel size, which is useful in smoothing out local optima, as elaborated in an earlier discussion.

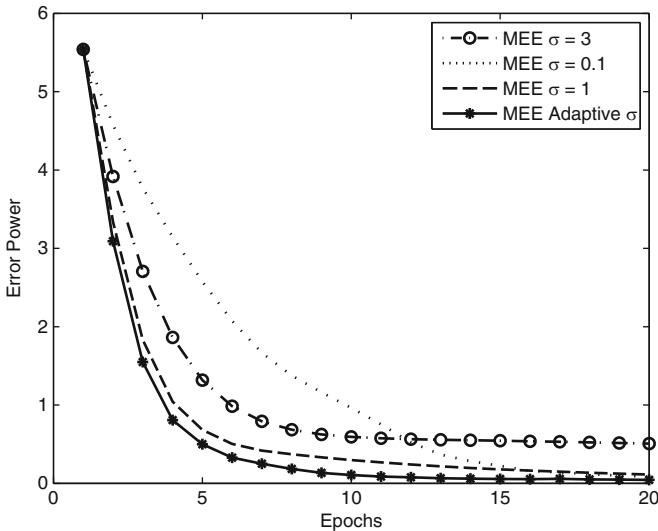
To quantify the advantage of an adaptive kernel width over fixed ones, we compare how the prediction error evolves during training for different values of kernel size. Figure 5.20 shows the error MSE (averaged over 200 Monte Carlo runs) over the 20 epochs, for three different fixed kernel sizes, 0.1, 1, 3, and also using the adaptive kernel.

As discussed, a large kernel size such as  $\sigma = 3$ , results in a high gradient, and therefore faster adaptation speed early on. However, as the optimal point approaches, the adaptation speed reduces, as seen from the plot. For a small kernel size such as  $\sigma = 0.1$ , there is relatively lesser gradient initially, but the rate of convergence is comparatively higher as the error power reduces, as can be seen in Figure 5.20. Because the adaptive kernel starts initially with a high kernel width, and an annealing of the kernel as the error reduces, a higher overall convergence rate is attained, shown by the solid curve in the plot.

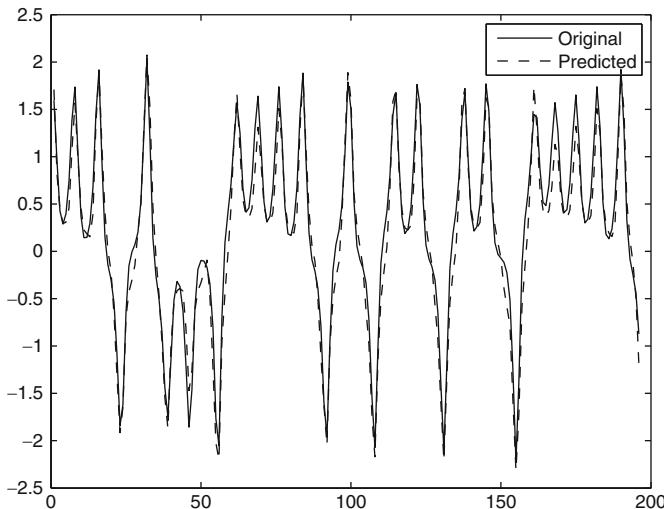
Figure 5.21 shows the prediction result from the last epoch of the training, superimposed on the original Lorenz attractor time series that was used in the experiment.

### System Tracking Using the MCC Algorithm

Our second experiment demonstrates the effectiveness of the proposed kernel adaptation method while identifying and tracking a time-varying linear sys-

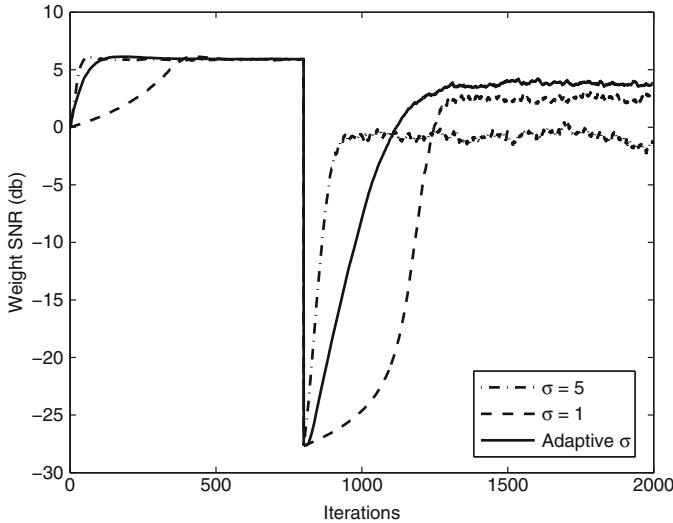


**Fig. 5.20.** Evolution of mean squared value of the error for different kernel widths, averaged over 200 Monte Carlo simulations. As expected, an adaptive kernel width has the highest rate of convergence, shown by the solid curve.



**Fig. 5.21.** Evolution of prediction results for the Lorenz time series for the trained TDNN with adaptive kernel size.

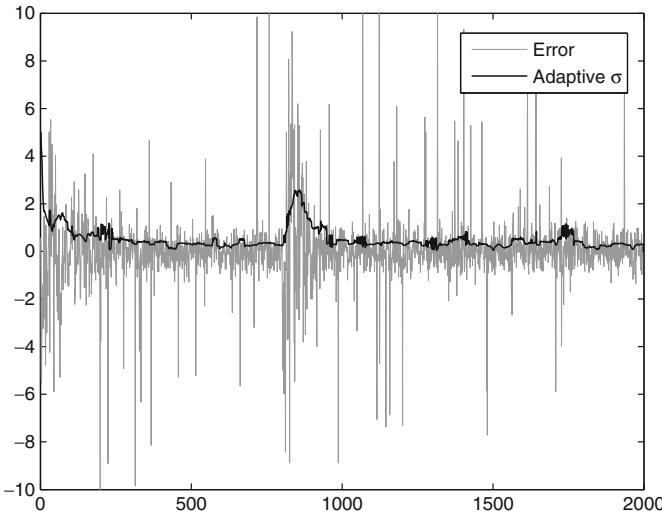
tem using the MCC algorithm in the presence of impulsive observation noise. The corresponding weight update equation, after using a stochastic approximation of the gradient is given as Eq. (4.63), which is strikingly similar to the LMS algorithm except for the Gaussian multiplicative factor evaluated



**Fig. 5.22.** Weight SNR learning curves while identifying and tracking a nonstationary system using the maximum correntropy criterion, with different kernel widths. The plots have been obtained after averaging over 200 Monte Carlo runs with different initializations.

at  $e(n)$ . Depending on the kernel width, the Gaussian evaluates to a small quantity if a large outlier error sample is encountered. This small value keeps the weight update stable, which would otherwise make a very large update if a strong outlier were encountered. Therefore, in the above weight update rule, having a small kernel size makes the algorithm more robust to impulsive noise. But, due to reasons mentioned earlier, a small kernel width also results in slower adaptation. We therefore expect that adapting the kernel size would lead to an optimal trade-off between these two factors.

For our simulation, we used a 4-tap adaptive filter to track another 4-tap filter. After 800 iterations, the original filter weights are changed from  $\mathbf{w}^* = [5, 0.1, 0.5, 0.3, 0.2]$  to  $\mathbf{w}^* = [0.4, 0.1, 0.5, 0.3, 0.2]$ . The input to the original filter and the adaptive filter is unit power Gaussian noise. The following impulsive noise is added as observation noise  $0.95G(0, 0.4) + 0.05G(0, 5)$ . Figure 5.22 shows the weight SNR (see Section 4.9) values for different choices of kernel sizes. As expected, a small kernel width such as  $\sigma = 1$  results in slower adaptation, but has higher WSNR values after convergence due to more robustness to impulsive noise. A large kernel width such as  $\sigma = 5$  causes faster adaptation, but loses the robustness to impulsive noise, and therefore has lower WSNR values after convergence. This is a clear tradeoff between speed and robustness for fixed values of kernel size. An adaptive kernel width, however, results in a better learning curve as it has better robustness and faster learning than  $\sigma = 1$ . It has a much higher WSNR value as compared to  $\sigma = 5$ .



**Fig. 5.23.** The error signal obtained while tracking using an adaptive kernel size in the cost function. The values of the kernel width over all the iterations are also shown.

Figure 5.23 shows how the kernel width adapts with the values of the error. When the filter weights are changed after 800 iterations, the errors become large. The kernel width adapts to this change and anneals again as the errors reduce. The increase in the kernel size improves the convergence speed, and the annealing results in more robustness to impulsive noise, after convergence.

## 5.9 Conclusions

This chapter shows how the EEC and ECC and in particular their MEE and MCC algorithms can be extended to nonlinear signal processing quite easily. The simplicity is based on the fact that EEC and ECC are differentiable costs, therefore the chain rule applies and the MEE/MCC can be easily brought into the backpropagation framework to train any nonlinear system (in particular neural networks) with gradient descent learning. In this respect the MCC is slightly simpler because the same code for BP can be used; we just need to change the injected error. We present the equations for backpropagation of the information forces as done in Chapter 2. We also present an overview of how to apply advanced search techniques to EEC, which is of practical importance for nonlinear systems in as much as gradient descent learning is commonly too slow and too brittle for nonlinear signal processing applications.

We included in the chapter four practical examples: the first in system identification, the second in channel equalization using decision feedback, both using the MEE, the third in linear and nonlinear regression and the fourth

showing the performance of the adaptive kernel size algorithm in system identification and tracking using both the MEE and the MCC algorithms to illustrate the fact that the same strategy works for both. These examples show performance comparisons with MSE and also provide a more detailed view of why MEE and MCC can be advantageous in practical problems. But they also point out that there is no universally better tool, and sometimes current thinking needs to be changed to explain the results (i.e. better equalization does not necessarily mean better error rates using the MEE criterion). The effect of  $\alpha$  in the performance of MEE has not been thoroughly investigated, but the equalization example shows that  $\alpha$  may play a role in practical problems.

One other conclusion is that the MCC is comparable to the MEE performance with a much reduced computational complexity, so it may show practical advantages in real-world scenarios. One issue to be resolved is the local minima in the cost function. Either kernel annealing or switching between MSE and ECC seem the best strategies.

Finally, the online adaptation of the kernel size seems to be an important addition to this set of cost functions, because it anneals the kernel size naturally during adaptation due to the decrease of error power. Although the cost function is proposed from the bias-variance dilemma perspective, in the examples tested it provides good overall adaptation characteristics; fast convergence with small final misadjustment. But ultimately, the kernel size evolution in adaptation depends upon the PDF of the error that is not under the control of the designer, so its use should be carefully tested in each practical scenario.

# Classification with EEC, Divergence Measures, and Error Bounds

Deniz Erdogmus, Dongxin Xu, and Kenneth Hild II

## 6.1 Introduction

The previous chapters provided extensive coverage of the error entropy criterion (EEC) especially in regard to minimization of the error entropy (MEE) for linear and nonlinear filtering (or regression) applications. However, the spectrum of engineering applications of adaptive systems is much broader than filtering or regression. Even looking at the subclass of supervised applications we have yet to deal with classification, which is an important application area for learning technologies. All of the practical ingredients are here to extend EEC to classification inasmuch as Chapter 5 covered the integration of EEC with the backpropagation algorithm (MEE-BP). Hence we have all the tools needed to train classifiers with MEE. We show that indeed this is the case and that the classifiers trained with MEE have performances normally better than MSE-trained classifiers. However, there are still no mathematical foundations to ascertain under what conditions EEC is optimal for classification, and further work is necessary.

The second aspect that we have yet to explore is the training of systems with divergence measures. In this chapter, we present learning algorithms based on the generalized information potential and the generalized information forces that implement the Euclidean and Cauchy-Schwarz divergence measures. In spite of the fact that divergence can train systems in any of the supervised or unsupervised learning frameworks, we concentrate in this chapter only on the supervised case and provide examples of its use in feature extraction and classification.

By extending Fano's bound with Renyi's entropy formalism, we can show that it is possible to bracket the probability of error in classification using Renyi's  $\alpha$  mutual information for  $\alpha < 1$  and  $> 1$ . This reinforces the argument that training a classifier with entropy measures should yield good classification performance.

## 6.2 Brief Review of Classification

The minimization of error entropy was motivated for regression in Chapter 3 from the point of view of minimizing the uncertainty in the error PDF. Is it possible to derive a similar solid framework for EEC in classification problems? Of course, it is always possible to use EEC as a cost function for classification, but this question goes beyond the engineering application and attempts to study the optimality of the criterion; that is, does the EEC yield a solution close to the minimum probability of error? It is therefore important to review briefly the theory of classification.

Consider the problem of assigning a pattern  $\mathbf{x}$  to one of  $L$  classes, represented by labels  $C_1, \dots, C_L$ . According to Bayes rule, the probability that  $\mathbf{x}$  belongs to class  $C_k$  is given by

$$P(C_k | \mathbf{x}) = \frac{P(C_k)p(\mathbf{x} | C_k)}{p(\mathbf{x})}, \quad (6.1)$$

where  $P(C_k | \mathbf{x})$  is the posterior probability of class  $C_k$  given that  $\mathbf{x}$  was observed,  $P(C_k)$  is the prior probability of class  $C_k$ ,  $p(\mathbf{x}|C_k)$  is the likelihood function and  $p(\mathbf{x})$  is the PDF of  $\mathbf{x}$  which plays the role of a normalization factor. According to Fisher [101] the decision that minimizes the classifier's error probability can be written as

$$\mathbf{x} \in C_k \quad \text{if} \quad k = \arg \max_{l=1, \dots, L} P(C_l | \mathbf{x}). \quad (6.2)$$

This became known as the maximum a posteriori (MAP) classifier. The decision rule provides a hypersurface in pattern space, effectively dividing the space in regions that are associated with each class. A proper discriminant function for class  $C_k$  is defined as any function that provides a large value for points  $\mathbf{x} \in C_k$  and low values for any other points in pattern space. The discriminant function plays a central role in classifier design because the class decision is made with respect to it. Effectively the optimal classifier uses the posterior density as the discriminant function, but any other function that preserves the class boundaries can be used instead. The wellknown linear and quadratic discriminant functions arise from Eq. (6.2) under a Gaussian assumption for the classes (with equal and different class covariance matrices, respectively) [80].

Instead of designing classifiers by fitting a Gaussian model to the likelihood function of the data (parametric design of classifiers), one can use nonlinear adaptive systems (commonly called neural networks) to construct the discriminant function. The classes of multilayer perceptrons (MLPs) or radial basis function (RBF) networks are commonly used because they are universal function approximators [38]. The nonparametric nature brings the need to find a figure of merit (cost function) to fit the model parameters to the training data.

The natural figure of merit in classification is the probability of error. However, this is hardly ever used as a cost function in classification because it is very difficult to evaluate in practice (effectively an integral over tails of class posteriors). Hence, researchers have used other costs as proxies for probability of error. Perhaps the most widely used method is to constrain the error power (MSE) between the output of the classifier and the correct targets in a training set [108].

Let us consider the usual classification problem where a pattern  $\mathbf{x} \in R^D$  is to be assigned to one of  $L$  classes by a learning machine (here we focus on a multilayer perceptron (MLP) with one hidden layer built from sigmoid nonlinearities and weight vector  $\mathbf{w}$ ). The MLP is trained using a set of training vector pairs  $\{(\mathbf{x}_i, \mathbf{c}_i), i = 1, \dots, N\}$ , where each target vector  $\mathbf{c}_i = [c_{i1}, \dots, c_{iL}]^T$  is of length  $L$  and describes the class to which  $\mathbf{x}_i$  belongs in a 1-out-of- $L$  coding, with  $c_{ij} \in [0, 1]$ . Hence, the MLP has an output layer described by a vector  $\mathbf{y}_i = [y_{i1}, \dots, y_{iL}]^T$  that produces for each  $\mathbf{x}_i$  its corresponding output  $\mathbf{y}_i$ . The mean square error (MSE) function

$$J_{MSE} = \frac{1}{N} \sum_{i=1}^N \|\mathbf{c}_i - \mathbf{y}_i\|^2 \quad (6.3)$$

is probably the most common error cost used for neural network training. Originally derived for regression, the MSE cost corresponds to applying the principle of maximum likelihood when the regression model is linear and the error is Gaussian distributed, zero-mean, and uncorrelated across exemplars. However, in classification the Gaussianity assumption on the error is invalid, due to the finite domain and multimodal nature of classification errors. It has been shown that classifier outputs trained with the MSE approximate the conditional average of the target data given the input, which under some conditions [38] means that the classifier outputs approximate the Bayesian posterior probabilities. However, in practice it is difficult to bound how far MSE training is from the minimum probability of error, and the simplicity of the procedure has been the main force behind its wide applicability. It is therefore important to look at alternative cost functions for classification.

### The Cross-Entropy Cost in Classification

Any differentiable function of the targets and outputs with a minimum for  $\mathbf{y} = \mathbf{c}$  (as does the MSE) is in principle an appropriate cost function for classification, but one generally seeks a choice based on solid statistical reasoning. One of the early alternatives that can also be derived from the maximum likelihood principle utilizes the crossentropy (CE) cost function [334]. Each component  $y_k$ ,  $k = 1, \dots, L$  of the output vector is interpreted as an estimate of the posterior probability that input pattern  $\mathbf{x}$  belongs to class  $C_k$  represented by  $c_k$ ,  $y_k = \hat{P}(C_k|\mathbf{x})$ , where the “true” marginal distributions of  $c$  given  $x$ ,  $\mathbf{p} = (p_1, \dots, p_L)$  are

$$p_k = P(C_k | \mathbf{x}), \quad k = 1, \dots, L \quad (6.4)$$

For mutually exclusive classes, the true conditional distribution  $p(\mathbf{c}|\mathbf{x})$ , and the MLP estimate  $\mathbf{y} = p_w(\mathbf{c}|\mathbf{x})$  (the dependence on the weights  $\mathbf{w}$  is made explicit) can be described by multinomial distributions:

$$p(\mathbf{c} | \mathbf{x}) = p_1^{c_1} p_2^{c_2} \cdots p_L^{c_L} \quad (6.5)$$

$$p_w(\mathbf{c} | \mathbf{x}) = y_1^{c_1} y_2^{c_2} \cdots y_L^{c_L} . \quad (6.6)$$

We would like the MLP output in Eq. (6.6) to approximate the true distribution Eq. (6.5). Considering a set of observed pairs  $\{\mathbf{x}_i, \mathbf{c}_i\}$  one can define the cross-entropy cost function between the true and estimated conditionals as

$$\begin{aligned} J_{CE} &= \log \frac{p(\mathbf{c} | \mathbf{x})}{p_w(\mathbf{c} | \mathbf{x})} = \log \prod_{i=1}^N \frac{p(\mathbf{c}_i | \mathbf{x}_i)}{p_w(\mathbf{c}_i | \mathbf{x}_i)} = \sum_{i=1}^N \log \left( \frac{p(\mathbf{c}_i | \mathbf{x}_i)}{p_w(\mathbf{c}_i | \mathbf{x}_i)} \right), \\ &= - \sum_{i=1}^N \sum_{k=1}^L c_{k,i} \log(y_{k,i}) + \sum_{i=1}^N \sum_{k=1}^L c_{k,i} \log(p_{k,i}) \end{aligned} \quad (6.7)$$

where we assume that the class labels are conditionally independent and  $\mathbf{c}_i$  depends only on  $\mathbf{x}_i$ . The goal is then to minimize Eq. (6.7) with respect to the weights;  $\min_w J_{CE}$ . Note that the second term does not depend on the MLP parameters  $\mathbf{w}$  which means that the minimization of Eq. (6.7) is equivalent to the minimization of

$$J_{CE} = - \sum_{i=1}^N \sum_{k=1}^L c_{k,i} \log(y_{k,i}). \quad (6.8)$$

Equation (6.8) is known in the literature as the cross-entropy cost function, because of its analogy with cross entropy ( $-E_c[\log(p(y))]$ ) in spite of the fact that  $c_{k,i}$  are not necessarily probabilities [145]. If we take the gradient of Eq. (6.8) with respect to the weights of a perceptron with tanh nonlinearities we find that the weight update becomes just dependent upon the error ( $e_k = c_k - y_k$ ) and independent of the derivative of the nonlinearity [145]. As we recall from Chapter 5, Eq. (5.2), the gradient of the MSE cost is multiplied by the derivative of the nonlinearity, which slows down training when the PE is even slightly saturated. Therefore we can expect much faster convergence with the cross-entropy cost when compared with MSE training. Experience has also shown that the CE cost normally works slightly better than the MSE for many problems [145].

### 6.3 Error Entropy Criterion in Classification

Another alternative is to use the EEC described in Chapters 3 and 4 that can be applied to both discrete and continuous error distributions and fully utilizes (for training) the shape of the error distribution. In Chapter 5 we saw how to

train nonlinear systems with EEC and its MEE algorithms, so the algorithm presented in Section 5.2 can be used directly here. However, classification is more complex than regression, so there are some important issues that need to be addressed to evaluate how appropriate EEC is for classification.

Silva et al. [298] have studied theoretically the EEC for the case of univariate data passing through a classifier with threshold nonlinearities that produce a discrete error random variable. For the special case of a two-class problem with a single parameter classifier this is equivalent to the Stoller split [309]. First, they proved that the EEC can only be related to the  $\min P(E)$  if the class priors are equal, which is a reasonable assumption in many practical problems (equal number of exemplars in each class). Furthermore, the  $\min P(E)$  always corresponds to a stationary point of the EEC cost but, depending upon the class separation, may correspond to either a minimum of entropy when the classes are well separated or to a maximum when the classes have large overlap. This is an unexpected result if we use our understanding from EEC in regression, and raises questions about the applicability of EEC for classification. It also does not corroborate experimental evidence that the EEC outperforms MSE in practical classification problems as these researchers and others have reported [86, 283].

In order to understand this difficulty, we take the continuous variable setting as in [299] because the discrete case is perhaps less appropriate to the information potential method described in this book, inasmuch as the error PMF is composed of only three values (e.g., for  $c = \{1, -1\}$  the possible errors occur at  $(-2, 0, 2)$  with different magnitudes). Let us consider a two-class problem from one dimensional data  $x$  with support  $D_X$  to be classified in two classes  $C_{-1}$  and  $C_1$ , using targets  $c = \{-1, +1\}$ , and a classifier producing continuous outputs  $y$ . We will assume that the classifier performs a mapping  $y = \varphi_w(x), y \in D_Y$  that depends upon the parameters  $w$  and the PE nonlinearity  $\varphi(x) = \tanh(x)$ . The PDF of the input can be written as

$$p_X(x) = \gamma p_{X|1}(x) + \beta p_{X|-1}(x), \quad x \in D_X \quad (6.9)$$

where  $\gamma$  and  $\beta$  are the prior class probabilities ( $\gamma = 1 - \beta$ ) and the notation  $p_{X|1}(x) \equiv p_X(x|c = 1)$  is used. So the error is a continuous random variable defined as  $E = C - \varphi_w(X)$ . In this setting it is not difficult to write the error PDF as

$$p(e) = \gamma p_{Y|1}(1 - e) + \beta p_{Y|-1}(-1 - e), \quad e \in [-2, 2]. \quad (6.10)$$

We present here just the simplest of the cases (see [299] for a complete treatment), with a single-parameter perceptron; that is,  $y = \tanh(x - w_0)$  where  $w_0$  is the bias. Renyi's entropy of the output error can be decomposed as

$$\begin{aligned} H_\alpha &= \frac{1}{1 - \alpha} \log \int_{-\infty}^{\infty} p^\alpha(e) de \\ &= \frac{1}{1 - \alpha} \log \left[ \int_{-2}^0 (\beta p_{Y|-1}(-1 - e))^\alpha de + \int_0^2 (\gamma p_{Y|1}(1 - e))^\alpha de \right]. \end{aligned} \quad (6.11)$$

Notice that due to the nonoverlapping domains in the two integrals, the information potentials are effectively separable and the minimization of Eq. (6.11) is equivalent to the maximization of the sum of the information potentials for  $\alpha > 1$  which is much simpler. For  $\alpha = 2$  we get

$$V_2(E) = \int_{-2}^0 (\beta p_{Y|1}(-1 - e))^2 de + \int_0^2 (\gamma p_{Y|1}(1 - e))^2 de. \quad (6.12)$$

Before actually presenting the result, let us compare the form of this  $V(E)$  with the regression case. We see that now there are two data distributions, one for each class and therefore, the distribution of the error is conditioned not only on the input data as it was in regression, but also on the class label  $C$ . It is still possible to evaluate Eq. (6.12) in closed form for simple densities such as the uniform and for the tanh nonlinearity (that has a range between  $[-1, 1]$ ). Assume we have two input overlapping classes, one uniform over  $[a, b]$  (class  $C_{-1}$ ) and the other over  $[c, d]$  (class  $C_1$ ) with  $a < c < b < d$ . Then the information potential of the error for the one-parameter perceptron is

$$\begin{aligned} V_2(E) = & -\frac{\beta^2}{4} \left[ \frac{2 + e(e+2) \log(|e|/(2+e)) + 2e}{(b-a)^2(2+e)e} \right]_{-1-\tanh(b-w_0)}^{1-\tanh(a-w_0)} \\ & + \frac{\gamma^2}{4} \left[ \frac{2 + e(e-2) \log(|e|/(e-2)) - 2e}{(d-c)^2(e-2)e} \right]_{1-\tanh(d-w_0)}^{1-\tanh(c-w_0)}. \end{aligned} \quad (6.13)$$

Silva [299] showed theoretical results with Eq. (6.13) that contradict the principle of minimization of the error entropy when the two classes overlap sufficiently, in the sense that the theoretical minimum probability of error occurs at the maximum of the error entropy for the one-parameter perceptron ( $w_0$ ). Likewise for the case of a two-parameter perceptron, there are many cases of poor class separability.

Perhaps the biggest conundrum is to conciliate these theoretical results with the experience that in practical problems (see Section 6.3), the classification results with EEC are often better than the same classifier trained with MSE. First and foremost, we have to realize that we do not train MLPs in the same way as implied by the theoretical analysis of Eq. (6.12). Indeed this cost function does not implement discriminative training because the errors are computed independently for each class, while in MLP training with EEC the error is computed in batch mode summing contributions of errors across classes. Second, one uses kernel estimation. In fact using Eq. (5.5), the information potential  $\hat{V}_2(E)$  can be written in the following way,

$$\begin{aligned}
\hat{V}(E) &= c \sum_{i \in C_{-1}} \sum_{j \in C_{-1}} G(e_i - e_j) + c \sum_{i \in C_1} \sum_{j \in C_1} G(e_i - e_j) \\
&\quad + 2c \sum_{i \in C_{-1}} \sum_{j \in C_1} G(e_i - e_j) \\
&= \beta^2 \hat{V}_{2,E|-1}(E) + \gamma^2 \hat{V}_{2,E|1}(E) + 2\beta\gamma \frac{1}{N_{-1}N_1} \sum_{i \in S_{-1}} \sum_{j \in S_1} G(e_i - e_j),
\end{aligned} \tag{6.14}$$

where  $c$  is a normalizing constant for the number of samples in each class,  $N_{-1}$  and  $N_1$  are the number of exemplars having class  $-1$  and  $1$ , respectively, and  $S_{-1}$  and  $S_1$  are the set of indices corresponding to class  $-1$  and  $1$ , respectively.

Just by looking at Eq. (6.14) we see that the IP estimator differs from the theoretical calculations of Eq. (6.12) because there is an extra term (the last term) that effectively computes the interactions between the errors of the two classes. For large kernel sizes this third term may be sufficiently large to change the cost function landscape. Silva [299] has shown that in some of the cases where the theoretical probability of error corresponds to a maximum of the entropy, the function estimated by Eq. (6.14) corresponds to a minimum, more in line with what we would expect from the EEC principle. Therefore, the conclusion is that the information potential of the error in batch mode is richer as a cost function for classification than the theoretical calculation of Eq. (6.12) which involves only the individual class entropies.

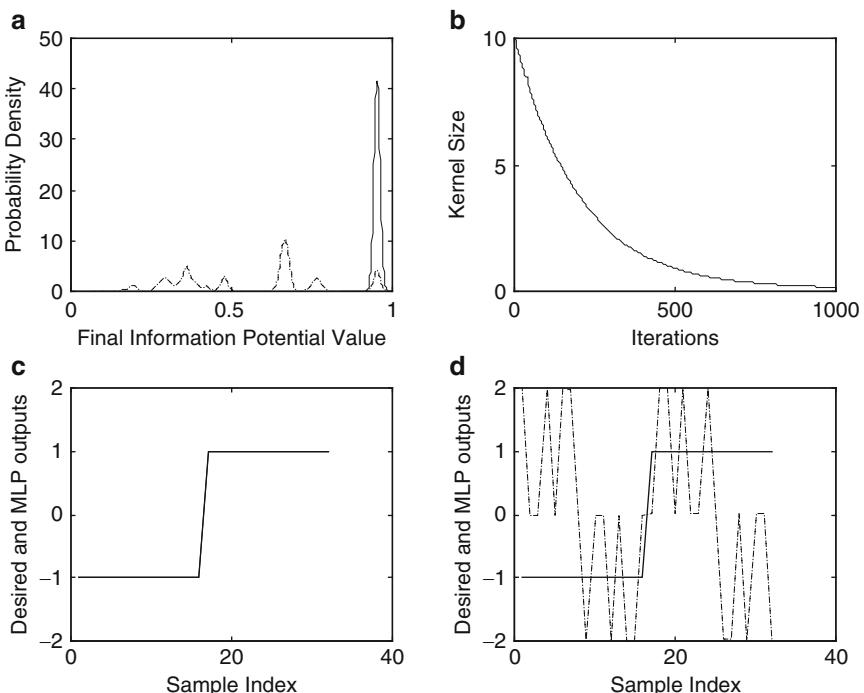
If we recall the estimation of the Euclidean or Cauchy–Schwarz divergences in Chapter 2 and apply it in a classifier setting, the cross-information potential is responsible for estimating the distance between the target and the classifier output densities. Notice that the third term in Eq. (6.14) can also be interpreted as a cross-information potential but now between the errors of each class. Although there are marked differences between the two expressions (Eq. (6.14) applies to errors, whereas the cross-information potential works directly with the density of targets and classifier outputs), this analogy deserves a more in-depth study to understand the difference between the theoretical EEC results and the practical applications of MEE, but it will not be pursued here.

### Performance of Classifier Trained with MEE

We illustrate here some of the published results of the application of EEC and its MEE algorithm in classification which have generally yielded better results than classification with the MSE cost function. However, we are not currently in a position to state under what conditions EEC is a better cost in classification so proper validation is required.

## Generalized Exclusive OR

We start with a simple example that shows the importance of kernel smoothing in classification. The generalized XOR problem is designed to provide an experimental demonstration of the global optimization property of the entropy-training algorithm we propose. Namely, the 5-bit parity problem in the class of generalized XOR problems is considered. In this case study, a 5-5-1 MLP with  $\tanh$  nonlinearities in the hidden layer and a linear output PE is used. The five inputs take the values  $\pm 1$  according to the considered bit sequence and the desired output is also  $\pm 1$ , corresponding to the case of even (+1) or odd (-1) number of ones. The training set consists of all possible 32 input sequences. In the constant kernel case, the kernel size is set to  $\sigma = 10^{-1}$  and the MLP is trained for 1000 iterations starting from 100 random initial weight vectors. In the annealed kernel case, the kernel size is annealed down exponentially as  $\sigma = 10 \rightarrow 10^{-1}$  in 1000 iterations for the same initial conditions (these values were experimentally obtained). The results of these experiments are summarized in Figure 6.1.



**Fig. 6.1.** Results for the XOR problem: (a) estimated probability densities of the final information potential values for the static kernel (dotted) and the annealed kernel (solid) cases; (b) annealing of the kernel size versus iterations; (c) annealed kernel case, desired output (solid), MLP output (dotted) (perfect match); (d) local optimum from the static kernel case, desired output (solid), MLP output (dotted) (from [86]).

In Figure 6.1a, the probability distribution of the final (normalized) information potential values is presented. Clearly, with the annealed kernels, the final information potential values are concentrated around the global maximum, whereas with the static kernels the algorithm is often trapped at local maxima. Figure 6.1b shows how the kernel size is exponentially annealed down in 1000 iterations. Figures 6.1c and d show optimal MLP outputs, one that exactly matches the desired output and a local optimum that produces a low-grade output. Therefore, the MLP trained using the annealed kernels achieved global optimum in all trials (100% of the time), whereas the MLP trained using the static kernels could only achieve the global optimum 10% of the time.

### MEE in Real Word Classification Tasks

The EEC criterion was also used to train single hidden layer MLPs with a different number of hidden units in the Diabetes, Wine, and Iris datasets of the UC Irvine repository by Santos, [283]. Each MLP was trained 20 times for 150 epochs with two fold cross-validation, both for MEE and MSE. The number of patterns for each of the databases is, respectively, 768, 178, 150. Santos et al. suggest an empiric formula to choose the kernel size,  $\sigma = 25\sqrt{L/N}$  where  $L$  is the dimension of the classifier output and  $N$  the number of samples. This value is normally much larger than the one found by density estimation formulas (e.g., Silverman's rule), and the author states that it consistently outperformed the Silverman's value in a large number of datasets (Ionosphere, Sonar, WDBC, IRIS, Wine, 2 Vowels PB, Olive). This result may be related to the discussion surrounding Eq. (6.14). Table 6.1 shows the results on the test set as a function of the number of hidden processing elements (PE).

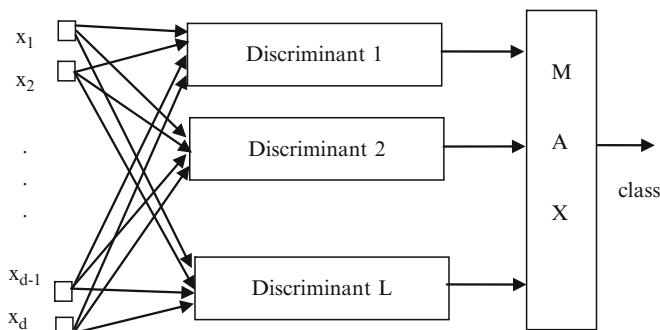
**Table 6.1.** Average Error Counts with MEE/MSE (From [283])

# PE	Diabetes MEE	Diabetes MSE	Wine MEE	Wine MSE	Iris MEE	Iris MSE
2	23.89	28.40	3.62	9.72	4.36	4.72
3	23.94	27.25	3.81	4.27	4.43	4.75
4	23.99	26.42	<b>1.94</b>	3.03	4.38	4.15
5	23.80	25.10	2.54	3.20	4.30	<b>3.97</b>
6	24.10	24.70	2.47	3.06	4.42	5.18
7	24.10	24.40	2.44	2.39	4.31	4.65
8	23.90	23.90	2.16	2.92		
9	24.30	24.00	2.22	2.50		
10	<b>23.60</b>	24.10	2.31	2.95		
11	24.02	27.41				
12	24.93	27.64				
STD	0.35	1.69	0.65	2.29	0.05	0.44

The authors concluded that the MLP trained with MEE has a more consistent performance across topologies; it seems to be less sensitive to the dimension of the hidden layer, and tends to have the best performance with fewer PEs. Finally, its performance also compares favorably with the MSE in two out of the three cases.

## 6.4 Nonparametric Classifiers

Neural networks are still parametric mappers (i.e., universal topologies with free parameters), but they are nonparametrically trained by means of a cost function; that is, one does not need generative models of the data to configure the free parameters to the problem at hand. This section addresses still another class of classifiers that works directly with the data to make classification decisions and that we appropriately call *nonparametric classifiers*. The PDF and the concepts of Renyi's  $\alpha$ -norm, information potential, and information forces outlined in Chapter 2 are utilized directly for classification. In fact, as we stated in Section 6.2, a valid discriminant function is any function that provides a large value for the partition of the space where the samples from the class reside and low values for any other part of the space. When the data classes do not overlap significantly, the PDF itself for each class estimated in a training set is such a function because it measures the density of samples in a region of the space. When class conditional PDFs are used as the discriminant functions then classification is performed using the maximum likelihood principle. Given an unlabeled sample in a test set, the maximum likelihood (ML) principle can decide which class it belongs to and therefore perform classification. A simple max operator is sufficient to implement the ML principle for many classes because partnership is discrete (Figure 6.2). The problem is that this method may not be very accurate in high-dimensional spaces, particularly when data are scarce; therefore it is only applicable to a subset of the cases of practical importance.



**Fig. 6.2.** General classifier built from discriminant functions.

However, it is very simple because it yields a nonparametric classifier; that is, one does not need a parametric system (neural network) to be trained to construct the appropriate discriminant function. We propose to utilize the  $\alpha$ -information potential that are intrinsically related to Renyi's  $\alpha$  entropies, so we are indirectly using an information theoretic classifier. With the  $\alpha$ -information potential we have two parameters to be set from the training set: the kernel size  $\sigma$  and the  $\alpha$  norm. Alternatively, two approaches are also possible that extend the range of applications for short datasets and/or high dimensional data: estimating the change in class divergence to include discriminative information; and projecting the data to a subspace that preserves discriminability with respect to the class labels, and then deciding class partnership based on the max operator (maximum likelihood).

### Information Potential and Force in Classification

Substituting Bayes rule into Eq. (6.2) we get

$$\mathbf{x}_0 \in C_k \quad \text{if} \quad k = \arg \max_{l=1,\dots,L} P(C_l)P(\mathbf{x}_0|C_l),$$

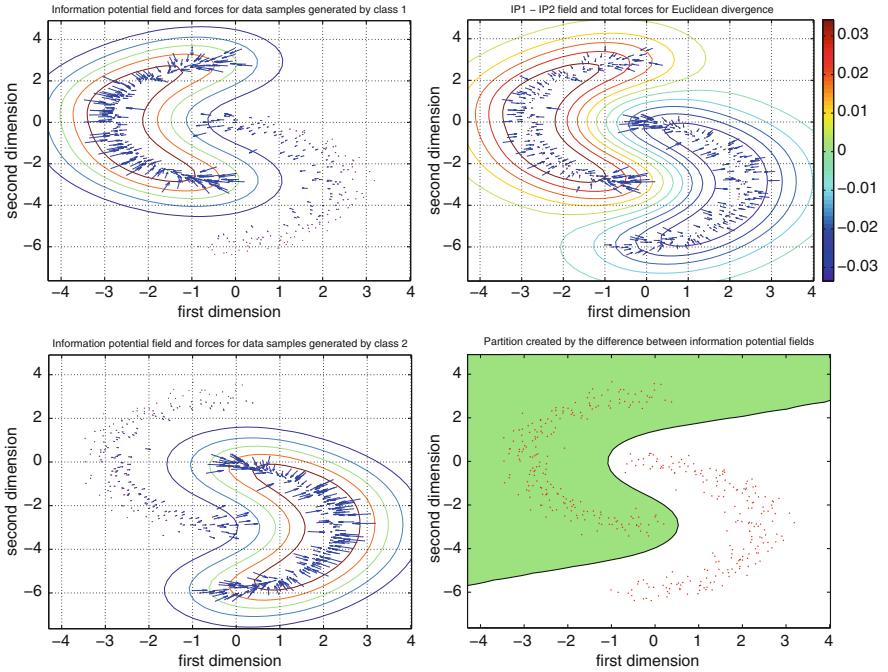
where we disregard  $p(x)$  because it is common to all classes. If we assume that the a priori probabilities are the same across the classes, the MAP rule for classification defaults to a test of likelihoods (i.e.,  $\mathbf{x}_0 \in C_k$  if  $k = \arg \max_l P(\mathbf{x}_0|C_l)$ ). This rule is very simple to implement with labeled data and Parzen estimators (the information potential field of Chapter 2). In fact, if we define  $\hat{p}(x|C_k) = 1/N_k \sum_{x_j \in C_k} \kappa(x - x_j)$ , the test of likelihood becomes

$$\mathbf{x}_0 \in C_k \quad \text{if} \quad k = \arg \max_{l=1,\dots,L} \hat{p}(\mathbf{x}_0|C_l) \tag{6.15}$$

That is, we estimate the probability density at  $p(\mathbf{x} = \mathbf{x}_0|C_1)$  from all the  $L$  likelihood functions and attribute the class partnership to the one that has the highest value. Figure 6.3 shows a simple 2D example of this rule for the two-class half-moon data where the decision rule defaults to  $\text{sign}(\hat{p}(\mathbf{x}|C_1) - \hat{p}(\mathbf{x}|C_2))$ . The kernel is a Gaussian with  $\sigma = 1$  and  $\alpha = 2$ .

Figure 6.3 also depicts the information forces to illustrate the “attraction” force created in pattern space due to the simple existence of the samples (information particles). In particular, notice the forces that are exerted at the tips of the half-moons that “attract” these samples to each cluster of samples. This is where the kernel size is critical because an inappropriate value may reverse the direction of the forces and then the classification for the samples at the boundary will be wrong. Moreover, the adaptation of the kernel size presented in Chapter 3, and illustrated in Chapter 5 can be readily applied here to fine tune the kernel size in the training set.

If one recalls the discussions in Chapter 2, the order  $\alpha$  of the norm also affects the forces because it weights differently regions of higher sample density versus regions of lower sample density dependent upon the  $\alpha$  selected. Indeed,



**Fig. 6.3.** The information potential field (IP) and the information forces for each class (left panel), the combined IPF and the boundary defined by the maximum likelihood rule (right panel).

from Eq. (2.68) we can conclude that the  $\alpha$ -IP field is built at every sample using a weighted Parzen estimation, where the weights are automatically defined by the density of samples in the neighborhood;

$$\hat{V}_\alpha(\mathbf{x}_j) = \hat{p}^{\alpha-2}(\mathbf{x}_j) \hat{V}_2(\mathbf{x}_j) = \frac{1}{N} \sum_{i=1}^N w_j \kappa_\sigma(\mathbf{x}_i - \mathbf{x}_j) \quad w_j = \hat{p}^{\alpha-2}(\mathbf{x}_j) \quad (6.16)$$

Equation (6.16) can be used in the training set to find out which value of  $\alpha$  produces the best results. In order to be less sensitive to the scarcity of the data, one can pose the classification problem in terms of divergences. Assume first a two-class problem, and that  $\hat{f}(x)$  and  $\hat{g}(x)$  are, respectively, the PDF estimates for class 1 and class 2 obtained from a training set. For a new test sample  $x_t$ , we can estimate the incremental change in divergence when  $x_t$  is placed in  $\hat{f}(x)$  or in  $\hat{g}(x)$ , denoted as  $\hat{f}^t(x), \hat{g}^t(x)$ , respectively. For instance, according to the Euclidean distance, we should place

$$x_t \in c_1 : \int (\hat{f}^t(x) - \hat{g}(x))^2 dx \geq \int (\hat{f}(x) - \hat{g}^t(x))^2 dx. \quad (6.17)$$

In words, we should place the sample in the class that most increases the divergence between the two classes (for mutual information the argument

is that the sample should be placed in the class that most decreases the mutual information). We show in Chapter 9 that this argument is also related to support vector machine classification. Moreover, the computation can be easily carried out with the IP of Chapter 2, and indeed without having to do much computation if the recursive information potential of Eq. (4.4) is utilized.

## 6.5 Classification with Information Divergences

Instead of training neural networks with MSE, EEC, or using the PDF as the discriminant function in nonparametric classification, one can use the concept of information divergence in classification, which includes the KL divergence, Euclidean and Cauchy–Schwarz divergences, as well as Shannon mutual information and the two quadratic mutual information algorithms defined in Chapter 2. In fact, the Fano bound is a strong theoretical result which shows that the probability of error in classification is lower bounded by the mutual information between data classes [97]. This link between mutual information and classification error means that mutual information is able to quantify separability between classes and suggests that it may be a very appropriate choice as a cost function for classification to improve upon the bound. The ITL techniques discussed in this book are readily applicable here as we have briefly mentioned in Eq. (6.17), which essentially is using the conditional entropy to implement a nonparametric classifier. Finally, the concept of information divergence also leads to the very interesting topic of discriminative projections that we frame as information filtering. These topics are expanded below.

### Discriminative Projections

The problem we treat in this section has wide applicability in pattern recognition when classifiers need to be implemented in large data spaces and the data are scarce. We present a methodology to adapt linear or nonlinear projection networks that find the most discriminative subspace for classification, taking full advantage of the fact that the labels in classification are indicator functions; that is, they are discrete random variables [340].

$X$  denotes a continuous multivariate random variable for the input data, and  $C$  the class label vector which is discrete. We are given a set of training data and their corresponding class labels  $\{(\mathbf{x}_i, \mathbf{c}_i), i = 1, \dots, N\}$ , and the task is to design a classifier that assigns the correct class reliably when given an unseen input  $\mathbf{x}$ . Again, classification can be formulated as a MAP problem as in Eq. (6.2):

$$\hat{c} = \arg \max_i P(C_i | \mathbf{x}) = \arg \max_c p_{XC}(\mathbf{x}, \mathbf{c}),$$

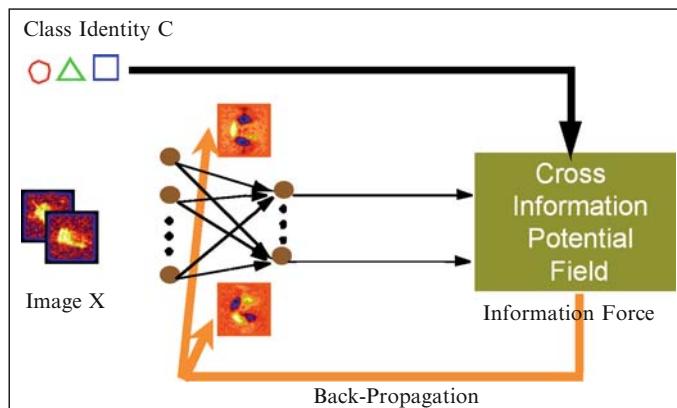
where  $P(C_i | \mathbf{x})$  is the a posteriori probability of the class label  $\mathbf{c}$  given the data  $\mathbf{x}$ , and  $p_{XC}(\mathbf{x}, \mathbf{c})$  is their joint PDF. Normally, a neural network or other

nonlinear system is trained to estimate directly the posteriors  $P(C_i|x)$ , but conceptually one can also estimate the joint PDF  $p_{\mathbf{x}, \mathbf{c}}(\mathbf{x}, \mathbf{c})$ . However, when the data  $\mathbf{x}$  are high-dimensional and scarce, it is often difficult to obtain a reliable estimation of the joint PDF. Dimensionality reduction (or feature extraction) is therefore appropriate. A multiple-input multiple-output (MIMO) *information filter*  $\mathbf{y} = f(\mathbf{x}, \mathbf{w})$  (where  $\mathbf{w}$  is a parameter set) is needed with  $\dim(\mathbf{y}) \ll \dim(\mathbf{x})$  such that its outputs  $\mathbf{y}$  can convey the most information about the class label and discard all the other irrelevant information. To be practical the dimensionality of  $\mathbf{y}$  should also be decoupled from the number of classes  $\mathbf{c}$ , so solutions that create an error by subtraction (as MSE or EEC) are inappropriate. The information filter outputs define a space for classification, but the difficulty is to design a method that projects the data and preserves discriminability.

Subspace projection is conventionally implemented with PCA (principal component analysis) due to its simplicity, but PCA does not preserve discriminability among classes [108]. Linear discriminant analysis (LDA) is an alternative, but it is restricted to linear projections and Gaussian classes so it suffers from these limitations [80]. The method outlined here seeks a projection that maximizes the mutual information of the projector output with the targets to preserve as much discriminability as possible in the reduced subspace, followed by a simple classifier (Figure 6.4). Based on the feature vector  $\mathbf{y}$ , the classification problem can be reformulated by the same MAP strategy of Section 6.4 as

$$\hat{c} = \arg \max_c p_{YC}(\mathbf{y}, \mathbf{c}) \quad \mathbf{y} = f(\mathbf{x}, \mathbf{w}), \quad (6.18)$$

where  $p(\mathbf{y}, \mathbf{c})$  is the joint PDF of the classification feature  $\mathbf{y}$  and  $\mathbf{c}$  the class label. The performance for this classification scheme crucially depends on how good the feature vector  $\mathbf{y}$  really is. The problem of reliable PDF estimation in



**Fig. 6.4.** Training a classifier with QMI<sub>ED</sub> (from [349]).

a high-dimensional space is now converted to the problem of building a reliable information filter for classification based only on the given training dataset. This goal is achieved again with the mutual information and the problem of finding an optimal information filter is formulated as

$$\mathbf{w}^* = \arg \max_{\mathbf{w}} I(Y = f(X, \mathbf{w}), C), \quad (6.19)$$

that is, to find the optimal parameter set  $w^*$  such that the mutual information between the classification feature  $Y$  and the class identity  $C$  is maximized. Of course the two optimization steps can be combined. To implement this idea, mutual information should be estimated nonparametrically to avoid restrictive assumptions, and this estimation is now done in the projected space, which is simpler. When there is a large difference in dimensionality between  $X$  and  $Y$  the information filter will be a simple linear projector although the method is also applicable with minor modifications to nonlinear mappers.

Notice that one of the important aspects of this formulation is that the dimensionality of  $\mathbf{y}$  is not restricted to be equal to the dimensionality of  $\mathbf{c}$  because Eq. (6.19) is working with the joint and each of the marginal distributions. This also is an indication that we suddenly are independent of the actual dimensionality of the variables, and are only interested in their PDFs.

## 6.6 ITL Algorithms for Divergence and Mutual Information

It should be apparent to the attentive reader that the learning procedures of Chapters 4 and 5 can be easily modified to include the Cauchy-Schwarz or Euclidean divergences or the corresponding mutual information definitions introduced in Chapter 2 because they are defined as additive combinations of different information potential fields. The big difference is that the error is not constructed; instead, the class labels and the output of the classifier are directly utilized to change the system parameters to match the label statistics.

In the ITL class of cost functions  $\Gamma(X)$ , the general gradient form to train system parameters  $\mathbf{w}$  (implementing a smooth map) is

$$\nabla_k \hat{\Gamma}(U) = \left( \frac{\partial \hat{\Gamma}(U)}{\partial (u(i) - u(j))} \right) \left( \frac{\partial (u(i) - u(j))}{\partial w_k} \right), \quad (6.20)$$

which is calculated by the chain rule over the combination of the system topology and cost function. Recall that the partial derivative in the first bracket is the information force that depends on the specific cost function (the cost variable  $u$ ), and the second term is the partial derivative of  $u$  with respect to the parameters of the system and depends upon the topology. If we want to substitute the EEC cost function with any of the DED, Dcs, or QMI<sub>ED</sub> or QMI<sub>CS</sub> the modifications are restricted to the first term (which may or may

not imply a modification in the second term because by the chain rule they need to be compatible). To illustrate, let us compute the gradient for D<sub>ED</sub> (Eq. (2.87)).

Remember that all the potentials in D<sub>ED</sub> work with a pair of random variables, unlike the EEC that only works with a single (the error) variable. To demonstrate the method, let us assume that we have two functions  $f(x)$  the system output and the desired response  $g(x)$ . As shown in Eq. (2.99) the potential field of D<sub>ED</sub>,  $\hat{V}_{ED}$ , is a combination of three different fields created by the samples of  $f(x)$ , the samples of  $g(x)$ , and the crossinformation potential among them. The corresponding information potentials are written for the case of  $N$  samples and where  $f(i)$  and  $g(i)$  refer to samples from  $f(x)$  and  $g(x)$ , respectively, assuming an arbitrary kernel  $\kappa$ .

$$\begin{aligned}\hat{V}_{ED} = & \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \kappa(f(i) - f(j)) - \frac{2}{N^2} \sum_{i=1}^N \sum_{j=1}^N \kappa(f(i) - g(j)) \\ & + \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \kappa(g(i) - g(j))\end{aligned}$$

In order to adapt a system with  $\hat{V}_{ED}$ , the injected error (first term in Eq. (6.20)) will be the information forces, that is, the partial derivative of  $\hat{V}_{ED}$  with respect to the system output variable  $f(x)$ . Therefore we just need to substitute the numerator of the first partial derivative in Eq. (6.20) with  $\hat{V}_{ED}$  and noticing that  $g(x)$  is independent of the system weights this yields

$$\begin{aligned}\nabla_k \hat{V}_{ED} = & \left( \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \frac{\partial \kappa(f(i) - f(j))}{\partial (f(i) - f(j))} \left( \frac{\partial (f(i) - f(j))}{\partial w_k} \right) \right) \\ & - \left( \frac{2}{N^2} \sum_{i=1}^N \sum_{j=1}^N \frac{\partial \kappa(f(i) - g(j))}{\partial f(i)} \left( \frac{\partial f(i)}{\partial w_k} \right) \right).\end{aligned}\quad (6.21)$$

The minimization of the distance between  $f(x)$  and  $g(x)$  in linear systems for weight  $k$  at iteration  $n$  uses the update equation (move in the opposite direction of the gradient).

$$w_k(n) = w_k(n-1) - \eta \nabla_k \hat{V}_{ED}(n-1).\quad (6.22)$$

Similar equations can be written for Cauchy-Schwarz divergence and extended to nonlinear systems (see Section 5.2).

## Information Filtering with QMI

In this section, we implement with quadratic mutual information the projection procedure that ideally preserves class discriminability. We show how

to adapt linear and nonlinear systems with quadratic mutual information using the generalized information potential. As we saw in Chapter 2, the case of the QMI is a bit more detailed than the divergences because we now have three different information fields in the joint and marginal spaces. The quadratic mutual information based on Euclidean distance  $\text{QMI}_{\text{ED}}$  and its corresponding generalized information potential  $I_{\text{ED}}$  are employed to illustrate the procedure. No assumption is made on the data density and when there is a large difference in dimensionality between  $X$  and  $Y$  the information filter is a simple linear projector. The method is also applicable with minor modifications to nonlinear mappers (the term in the right bracket of Eq. (6.16) will have to be substituted by the delta rule or backpropagation algorithm).

In the case study that follows the input are images of size  $64 \times 64$  (4096 dimensions), and there are three classes in a two-dimensional space, so we use a two-dimensional projection space to derive the equations for  $\text{QMI}_{\text{ED}}$ . In classification, the three labels are discrete and define a probability mass function that can be written as  $p(\mathbf{c}) = N_1/N\delta(\mathbf{c} - \mathbf{c}_1) + N_2/N\delta(\mathbf{c} - \mathbf{c}_2) + N_3/N\delta(\mathbf{c} - \mathbf{c}_3)$  where  $N_i$  represents the cardinality of class  $i$  divided by the number of samples,  $\mathbf{c}_i$  are the values of the class labels and the delta function is defined as  $f(x_0) = \int \delta(x - x_0)f(x)dx$ . Recall that  $\text{QMI}_{\text{ED}}$  is specified by the generalized information potential that for this case is given by

$$\begin{aligned} I_{\text{ED}} &= \iiint (p_{YC}(\mathbf{y}, \mathbf{c}) - p_Y(\mathbf{y})p_C(\mathbf{c}))^2 dy_1 dy_2 d\mathbf{c} \\ &= \iiint p_{YC}(\mathbf{y}, \mathbf{c})^2 dy_1 dy_2 d\mathbf{c} - 2 \iiint p_{YC}(\mathbf{y}, \mathbf{c})p_Y(\mathbf{y})p_C(\mathbf{c})dy_1 dy_2 d\mathbf{c}, \\ &\quad + \iiint (p_Y(\mathbf{y})p_C(\mathbf{c}))^2 dy_1 dy_2 d\mathbf{c} \end{aligned} \quad (6.23)$$

where  $y_1$  and  $y_2$  are the two outputs and  $c$  is the class label. Because each term is a quadratic function of PDFs we can use the IP as the estimator, and obtain

$$\begin{aligned} \hat{V}_J &= \iiint \hat{p}_{YC}^2(\mathbf{y}, \mathbf{c})dy_1 dy_2 d\mathbf{c} \\ &= \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N [G_{\sigma\sqrt{2}}(y_1(i) - y_1(j)) \cdot G_{\sigma\sqrt{2}}(y_2(i) - y_2(j)) \cdot \delta_K(\mathbf{c}(i) - \mathbf{c}(j))] \end{aligned} \quad (6.24)$$

where  $\delta_K(\mathbf{c}) = 1$ , iff  $\mathbf{c} = 0$  is called the Kronecker delta (the discrete version of the delta function). Alternatively, one could have used the fact that labels are discrete and work directly with a mixed (discrete/continuous) joint density. For the other terms we have

$$\begin{aligned}\hat{V}_C &= \iiint \hat{p}_{YC}(\mathbf{y}, \mathbf{c}) \hat{p}_Y(\mathbf{y}) \hat{p}_C(\mathbf{c}) dy_1 dy_2 d\mathbf{c} \\ &= \frac{1}{N} \sum_{i=1}^N \left[ \left( \frac{1}{N} \sum_{j=1}^N G_{\sigma\sqrt{2}}(y_1(i) - y_1(j)) G_{\sigma\sqrt{2}}(y_2(i) - y_2(j)) \right) \right. \\ &\quad \left. \cdot \left( \frac{1}{N} \sum_{k=1}^N \delta_K(\mathbf{c}(i) - \mathbf{c}(k)) \right) \right] \end{aligned} \quad (6.25)$$

$$\begin{aligned}\hat{V}_M &= \iiint (\hat{p}_Y(\mathbf{y}) \hat{p}_C(\mathbf{c}))^2 dy_1 dy_2 d\mathbf{c} \\ &= \left( \frac{1}{N^2} \sum_{i=1}^N \sum_{k=1}^N G_{\sigma\sqrt{2}}(y_1(i) - y_1(k)) G_{\sigma\sqrt{2}}(y_2(i) - y_2(k)) \right) \\ &\quad \cdot \left( \frac{1}{N^2} \sum_{j=1}^N \sum_{m=1}^N \delta_K(\mathbf{c}(j) - \mathbf{c}(m)) \right). \end{aligned} \quad (6.26)$$

By noting the property of the delta function, the sums are nonzero only when  $i$  and  $j$  are drawn from the same class. So, the calculation of  $V_J$  defaults to the marginal PDF of  $Y$  estimated at the sample pairs that belong to the class, which saves computation. Assuming in general that there are  $L$  classes, that the samples are ordered from each class as  $N_1, N_2, \dots, N_L$ , and a vector representation for the outputs is utilized with an index that refers to the class (when needed) then Eq. (6.24)–(6.26) become

$$\begin{aligned}\hat{V}_J &= \frac{1}{N^2} \sum_{p=1}^L \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} [G_{\sigma\sqrt{2}I}(\mathbf{y}_p(i) - \mathbf{y}_p(j))] \\ \hat{V}_C &= \frac{1}{N^2} \sum_{p=1}^L \frac{N_p}{N} \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} G_{\sigma\sqrt{2}I}(\mathbf{y}_p(i) - \mathbf{y}_p(j)) \\ \hat{V}_M &= \frac{1}{N^2} \left( \sum_{p=1}^L \left( \frac{N_p}{N} \right)^2 \right) \sum_{i=1}^N \sum_{j=1}^N G_{\sigma\sqrt{2}I}(\mathbf{y}(i) - \mathbf{y}(j)). \end{aligned} \quad (6.27)$$

Once  $\hat{I}_{ED}$  is estimated, then we still need to adapt the projector parameters. The concept is exactly the same as explained in Section 6.6, with  $\hat{I}_{ED}$  substituting  $\Gamma$  in Eq. (6.20). Because we are interested in performing gradient ascent in the  $\hat{I}_{ED}$  field (maximizing mutual information), the weight update equation Eq. (6.22) uses a plus sign instead of the minus sign and  $\hat{V}_{ED}$  substituted by  $\hat{I}_{ED}$  as

$$w_k(n) = w_k(n-1) + \eta \nabla_k \hat{\mathbf{I}}_{ED}(n+1), \quad \hat{I}_{ED} = \hat{V}_J - 2\hat{V}_C + \hat{V}_M. \quad (6.28)$$

The QMI<sub>ED</sub> information force in this particular case can be interpreted as repulsion among the information particles with different class identity, and attraction among the information particles within the same class.

The appeal of this methodology is that once the projector to a small feature space is adapted, it is easy to estimate the joint PDF  $p_{YC}(y, c)$  by Parzen windowing because the dimensionality now is low, which immediately yields the MAP decision rule of Eq. (6.18). Moreover, the class label is a discrete variable, thus the search for the maximum in Eq. (6.19) can be simply implemented by comparing each value of  $p_{YC}(y, c)$  to find the maximum.

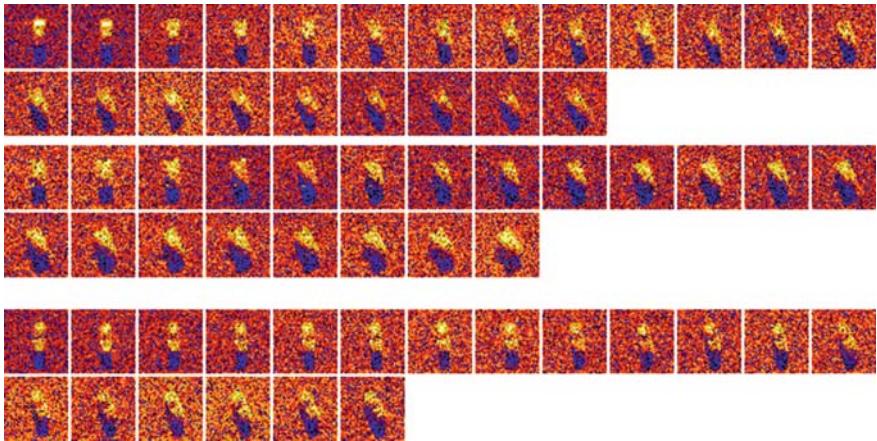
There is a free variable in this method that is the best dimension  $M$  of the subspace that obviously affects the quality of the final classification. One can adopt the concepts of dimensionality analysis in dynamical systems and modify them to our setting. Potentially, Fano's bound can be used to help select the projection space dimensionality (see Section 6.8). The information divergence formulation frees us from the use of EEC and allow us to truly start exploring cost functions that manipulate divergences in probability spaces which is the true appeal of information-theoretic learning (ITL). It is worth mentioning another interesting feature of the divergence-based cost functions. As noted in Chapter 1, the same criterion can be used for unsupervised learning, which is fully exploited in Chapter 7.

### 6.6.1 Case Study: Automatic Target Recognition (ATR) with ITL

Before we actually describe the data it is necessary to explain briefly the characteristics of automatic target recognition as open set classification. Automatic target recognition (ATR) is more demanding than the conventional classification problems where all the data belong to one of the trained classes (as illustrated in Section 6.3), which is normally called close set classification. When an ATR classifier is deployed in the real world, it is very likely that new vehicles (called confusers) will be encountered, that is, vehicles that belong to none of the vehicle classes used in the training set (open set classification). Therefore, the testing of ATR classifiers requires one or more confuser classes to verify their performance. The issue can be understood in terms of false positives (detection of confusers as belonging to one of the trained classes), and missed detections (i.e., to reject the confusers some of the test vehicles may not be classified because the confidence level of the classifier may be too low to make a decision).

The aspect angle of a target (i.e., the angle from which the target is viewed) is unknown, therefore ATR classifiers have to be trained for all 360 degrees of aspect angles. Because the SAR image of a target is based on radar reflections on the target, different aspect angles of the same target will result in quite different SAR signatures. Instead of training a single classifier for the full 360 degrees of aspect angle, several classifiers can be trained for different aspect angle sectors (this is a divide and conquer strategy), although it requires a good estimator for the aspect angle. In this divide and conquer approach better classification accuracy is obtained in this difficult problem [349].

Finally to evaluate better the generalization ability it is common practice to create a test set of the same vehicles that is taken at a different depression



**Fig. 6.5.** The SAR images of three vehicles for training classifier (0–30 degree) (from [349]).

angle (the angle between the sensor line of sight and ground) of the training set. This increases the difficulty of the task inasmuch as the radar returns in angular objects change drastically with the depression angle.

The experiment is conducted on the MSTAR database [324]. There are three classes of vehicles, BMP2, BTR70, and T72, classified from  $64 \times 64$  pixel synthetic aperture radar (SAR) images with the approach described in Section 6.6. The system diagram is shown in Figure 6.4.

Figure 6.5 shows the training images (22, 21, 19 images per class, respectively) to illustrate the difficulty of the task.

Here, two types of confusers were used, so the test set was built from images taken at a 15 degree depression angle of the following vehicles (for each class, there are some different configurations) as shown below.

BMP2——BMP2\_C21, BMP2\_9563, BMP2\_9566

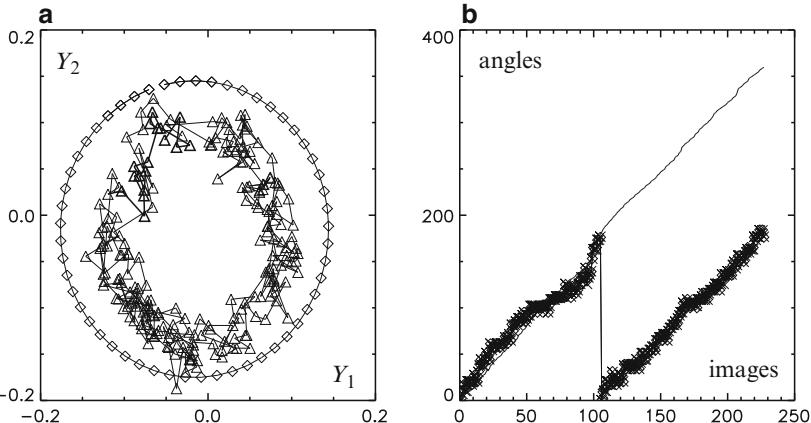
BTR70——BTR97\_C71

T72——T72\_132, T72\_S7, T72\_812

Confuser——2S1, D7

### Aspect Angle Estimation with QMI

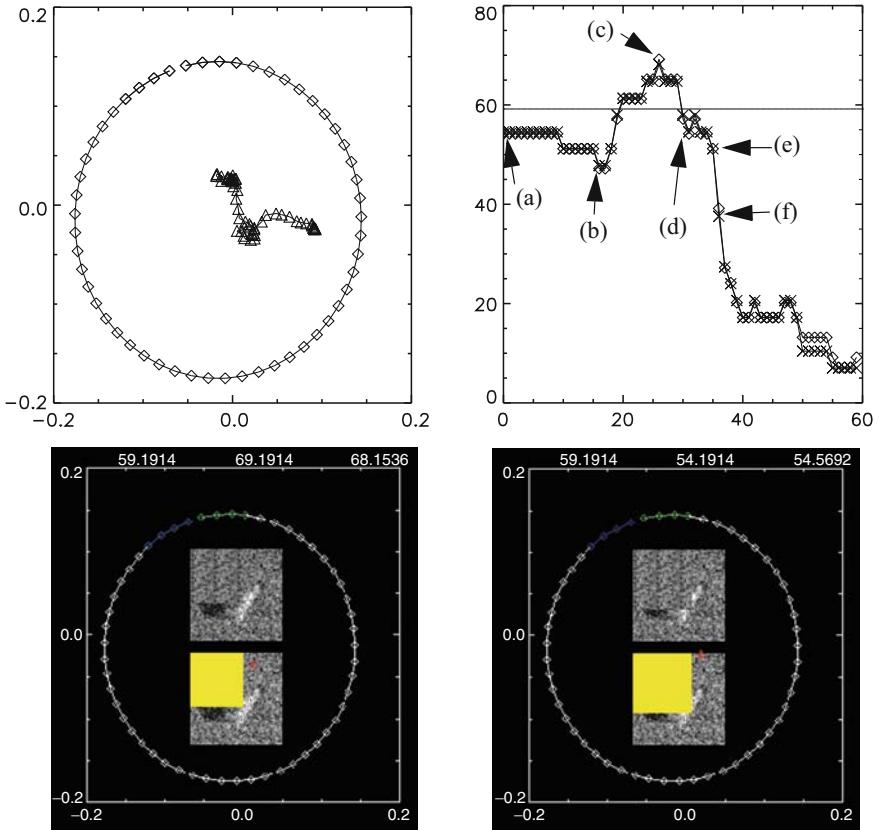
The design of the aspect angle estimator is done first to show the large applicability of this nonparametric MAP approach in pattern recognition applications. For instance, the problem of aspect angle (pose) in an image set can be framed as the following maximum a posteriori problem. From the input image  $\mathbf{x}$  find a projection  $\mathbf{y}$  that is maximally informative with respect to the unknown aspect angle  $\theta$  (the latency variable) of the vehicle in the image:  $\hat{\theta} = \arg \max_{\theta} p_{Y|\theta}(\mathbf{y}, \theta)$   $\mathbf{y} = f(\mathbf{x}, \mathbf{w})$ . For this particular case, because the pose



**Fig. 6.6.** Performance of the pose estimator based on  $\text{QMI}_{\text{ED}}$  when trained on a vehicle and tested in another. In (a), diamonds are the training set poses and the triangles the test set results. The axis in (b) shows the estimated angle (vertical) versus the test image number ordered by increasing aspect angle (from [340]).

angle is a one-dimensional periodic variable we code it as a two-dimensional vector  $\mathbf{a} = [a_1, a_2]$ , where  $a_1 = \sin(2\theta)$  and  $a_2 = \cos(2\theta)$  and construct a linear projector from 4096 dimensions to a 2D space of outputs  $[y_1, y_2]$  trained to maximize the  $\text{QMI}_{\text{ED}}$  between  $\mathbf{y}$  and  $\mathbf{a}$ . The difficulty of distinguishing between front and back views in the SAR images of targets led us to use the double of the angle to wrap around the angles between 0–180 instead of 0–360. Once the system is trained with 53 exemplars (3.5 degree resolution), a new image of a different vehicle is input to the system, the maximum value of  $\mathbf{y}$  is found, and the true pose is this angle divided by two to compensate for  $\mathbf{a}$ . The kernel size was selected at  $\sigma = 0.01$  (Silverman's rule) and a large range of kernel sizes (an order of magnitude in both directions) gave basically the same results. Figure 6.6 shows an example of the pose estimator performance when it is trained in one vehicle (BMP2) and tested in another (T72), while the vehicle is being rotated from 0 to 360 degrees.

The left panel with diamonds shows the training performance, and the triangles the test results. The right panel shows the pose estimation on the  $y$ -axis versus the image number (ordered by increasing value of the aspect angle) denoted as stars, and the true pose in the solid line. Due to the fact that we only trained for 0–180 degrees, there is a wraparound phenomenon with a jump of 180 degrees due to the confusion between front and back views. If this jump is discounted the average pose error is 6.18 degrees and the s.d. is 5.18 degrees. When the same network is trained with MSE the training set error is slightly smaller, but the test results are much worse. The big advantage of  $\text{QMI}_{\text{ED}}$  training versus the MSE was obtained when the image was occluded



**Fig. 6.7.** Occlusion results. Top row shows the overall results of the pose through progressive occlusions in polar coordinates (left) and in value (right). The bottom figures show two instances (c) and (e). Numbers on top represent (left to right) the true pose and two different pose estimators (ML or mean of estimated PDF) (from [340]).

with increasing amounts of background. As shown in Figure 6.7 the vehicle image can be mostly occluded but the pose estimator is still reasonable.

Figure 6.7 shows in the top left panel the changes of pose estimation across progressively larger occlusions. It is noticeable that the values (triangles) decrease towards the center of the circle but maintain more or less the same angle. In the top right panel a more detailed view of the estimation is provided, where the true pose is shown (the solid horizontal line at 60 degrees), and the estimations at different occlusions (horizontal axis). Two of the cases (c) and (e) are shown in the bottom two panels. It is remarkable that it is sufficient to observe a tiny part of the boundary of the object for the system to estimate its pose.

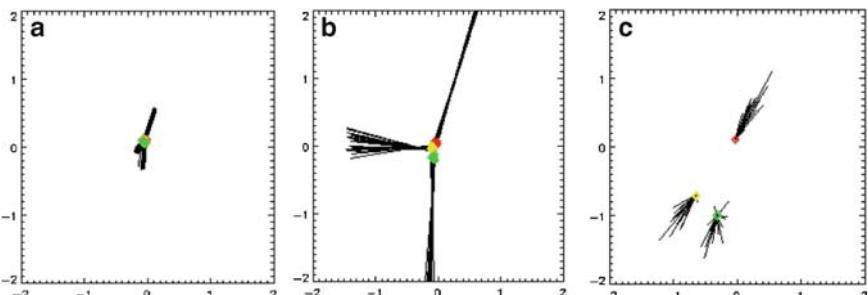
## Training the Projector for Classification

The training of the discriminability-preserving projection is accomplished in the same way as the pose estimator, except that now the labels are the three target classes. That is, a linear network with 4096 inputs is trained to maximize the mutual information between the output of the projector in 2D space and the class labels, using gradient ascent learning Eq. (6.28). Although there are three classes, a two-dimensional feature space gave very good results and is the one used for visualization. A training set built from 62 exemplars of the three classes within the 0–30 degree sector were used for training (Figure 6.5 depicts the images from each vehicle included in the training set). The kernel size was kept at the same value as for the pose estimator, the step size was  $\eta = 5 \times 10^{-5}$ , training was stopped at 200 iterations, and the class label positions were chosen as the vertices of an equilateral triangle embedded in a circumference of unit radius.

Snapshots of the network during training displaying the information forces acting on the samples of each class are depicted in Figure 6.8 at different phases of training. Panel A shows the initial phase with the three classes still mixed together, but the forces are already pointing outward; Panel B shows the result after 50 iterations with classes starting to separate, and the forces are very strong at this stage. Panel C shows the output data distribution at the final stage of the training where the three classes are clearly separated, each class tends to shrink to one point, and the forces decrease towards zero.

After the projector is trained, the class PDFs estimated with the information potential field (Eq. (6.16)) are used to assign the test set exemplars to the classes using the MAP rule. Table 6.2 shows the classification result for the sector 0–30 degrees. In spite of the limited number of training data, the classifier still shows very good generalization ability because only two errors occur in the test set.

For comparison purposes, an SVM classifier using the Gaussian kernel was trained with cross-validation for the slack variables for best performance [349].



**Fig. 6.8.** Information forces acting on the samples of each class during training (from [254]).

In this particular sector the SVM had higher errors (7 versus 2), but the overall error rate of both classifiers in all the quadrants was statistically equivalent and it is shown in Table 6.2.

The open set classification results for this sector with the two confuser vehicles were also established by setting a threshold on the probability of detection ( $P_d = 90\%$ ). Table 6.3 shows the detection performance for the classifier trained with ITL and SVM. ITL records four false positives for the two confusers, but SVM rejects them all. However, SVM misses more T 72 targets, so we conclude that the classification result of QMIED is only slightly worse than the support vector machine [349].

In terms of overall performance in all the sectors, Table 6.4 shows the overall error rate for the test set (724 samples) of the template matcher (which is the most widely used classifier for this problem using 36 templates per target), a multiresolution PCA projector [349], the QMIED classifier, and the SVM. As the table shows, the QMI and SVM have comparable performance,

**Table 6.2.** Confusion Matrices for the ITL and SVM Classification (from [349])

ITL	BMP2	BTR70	T72	SVM	BMP2	BTR70	T72
BMP2-C21	18	0	0	BMP2-C21	18	0	0
BMP2-9563	11	0	0	BMP2-9563	11	0	0
BMP2-9566	15	0	0	BMP2-9566	15	0	0
BTR70	0	17	0	BTR70	0	17	0
T72-132	0	0	18	T72-132	0	0	18
T72-812	0	2	9	T72-812	5	2	4
T72-S7	0	0	15	T72-S7	0	0	15

**Table 6.3.** Confusion Matrices for Detection ( $P_d = 0.9$ ) (from [349])

ITL	BMP2	BTR70	T72	Reject	SVM	BMP2	BTR70	T72	Reject
BMP2-C21	18	0	0	0	BMP2-C21	18	0	0	0
BMP2-9563	11	0	0	2	BMP2-9563	11	0	0	2
BMP2-9566	15	0	0	2	BMP2-9566	15	0	0	2
BTR70	0	17	0	0	BTR70	0	17	0	0
T72-132	0	0	18	0	T72-132	0	0	18	0
T72-812	0	2	9	7	T72-812	0	1	2	8
T72-S7	0	0	15	0	T72-S7	0	0	12	3
2S1	0	3	0	24	2S1	0	0	0	27
D7	0	1	0	14	D7	0	0	0	16

**Table 6.4.** Overall Classification Results (from [349])

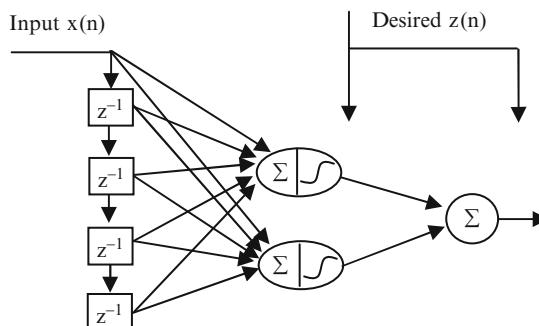
Classifier	Error Rate (%)
Template	9.60
PCA-M	10.3
QMI-ED	5.93
SVM	5.13

but they achieve classification in totally different ways: the SVM uses 400 support vectors to represent the classes, whereas the QMI<sub>ED</sub> chooses the best 2D projection that separates the classes.

This test with SAR ATR classification highlights the differences that we have mentioned between training with EEC and training with QMI. The training of the classifier really can be thought of as a projection to a lower-dimensional space that preserves discriminability followed by classification in the subspace. Hence, each one of the classes is mapped independently of the other, unlike the case of the EEC which combines the errors of each class and changes the weights with this combined information. We have only presented results where the IP is used for both projecting the data and classification, but any other classifier can be used in the projected space.

### Case Study: Training (deep) Networks Layer by Layer with QMI

The conventional paradigm to train multilayer networks is by backpropagating the errors across the network layers. This is a clever and well-studied procedure that brings error information from the top (i.e., closer to the output) layers back to the bottom (i.e., closer to the input) layers and trains the multilayer system in a cohesive way. However, in computational neuroscience, the biological plausibility of the error backpropagation step has been questioned, and in more general terms, it is conceptually important to know if a purely feedforward training procedure is able to discover complex mappings one stage at a time (e.g., deep and growing networks). We show here that maximizing quadratic mutual information (using QMI<sub>ED</sub>) at each layer of a nonlinear network (MLP) with respect to a desired signal is able to discover complex mappings. In a sense the maximization of the mutual information with respect to the training signal at each layer substitutes for the backpropagated error from the top layers. We basically use the methodology of Section 6.5 to find subspace projections. We consider each network layer as an information filter that is trained to convey in its output ( $Y$ ) as much input information ( $X$ ) about the desired response  $Z$  as possible (Figure 6.9).



**Fig. 6.9.** TDNN trained one stage a time without error backpropagation.

$$\mathbf{w}^* = \arg \max_{\mathbf{w}} I(Y = f(X, \mathbf{w}), Z).$$

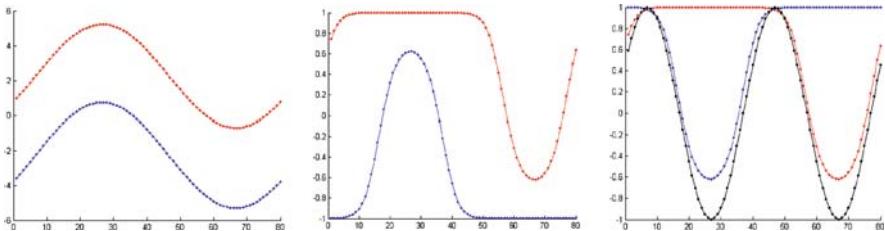
Equation (6.23) is the one that applies to QMI<sub>ED</sub>, except now the class labels  $c$  are substituted by the desired signal  $z(n)$  which is a continuous variable, so there are no simplifications in the kernel estimation of  $\hat{V}_J$ ,  $\hat{V}_c$ ,  $\hat{V}_M$  (see Eq. (2.103)). Because each layer of the MLP is nonlinear, Eq. (6.20) needs to be applied with an update for nonlinear PEs, which results in the following gradients (GIFs) for each one of the field components

$$\begin{aligned}\frac{\partial V_J}{\partial w} &= \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \left[ G_{\sqrt{2}\sigma} (y(i) - y(j)) \cdot G_{\sqrt{2}\sigma} (z(i) - z(j)) \right. \\ &\quad \left. \cdot \frac{-(y(i) - y(j))}{\sqrt{2}\sigma_y^2} \cdot (f'(i)x(i) - f'(j)x(j))^T \right] \\ \frac{\partial V_C}{\partial w} &= \frac{1}{N} \sum_{i=1}^N \left[ \left( \frac{1}{N} \sum_{j=1}^N G_{\sqrt{2}\sigma} (y(i) - y(j)) \cdot \frac{-(y(i) - y(j))}{\sqrt{2}\sigma_y^2} \right. \right. \\ &\quad \left. \left. \cdot \left( f'(i)x(i) - f'(j)x(j) \right)^T \right) \cdot \left( \frac{1}{N} \sum_{k=1}^N G_{\sqrt{2}\sigma} (z(i) - z(k)) \right) \right] \\ \frac{\partial V_M}{\partial w} &= \left( \frac{1}{N^2} \sum_{i=1}^N \sum_{k=1}^N G_{\sqrt{2}\sigma} (y(i) - y(k)) \cdot \frac{-(y(i) - y(k))}{\sqrt{2}\sigma_y^2} \right. \\ &\quad \left. \cdot (f'(i)x(i) - f'(k)x(k))^T \right) \cdot \left( \frac{1}{N^2} \sum_{j=1}^N \sum_{m=1}^N G_{\sqrt{2}\sigma} (z(j) - z(m)) \right).\end{aligned}$$

We demonstrate this training in the so-called frequency doubler, already mentioned in Chapter 5: a TDNN that duplicates the frequency of any periodic signal presented at its input (Figure 6.9).

A system that doubles the frequency of its input must be nonlinear because, by definition, linear systems do not create new frequencies. Here we construct a TDNN with four delays in the input layer, two nonlinear processing elements ( $\tanh$ ) in the hidden layer, with a single linear processing element in the output layer. The number of delays should be selected to create a proper embedding of the input signal dynamics (for a sinewave one single delay is sufficient, but more delays simplify the discovery of the mapping).

The input sinewave  $x(n)$  has 80 samples per cycle, and the desired response  $z(n)$  has 40 samples per cycle. QMI<sub>ED</sub> and gradient ascent learning with kernel annealing are used here as the criterion and adaptive algorithm to train the parameters of the TDNN, except that the training proceeds as follows. The input layer weights are trained first until convergence, are fixed, and only then the output weights are trained. Although it is possible to train all the weights at the same time, this procedure is wasteful because the output layer learns its part of the mapping only after the first layer training stabilizes. The kernel size

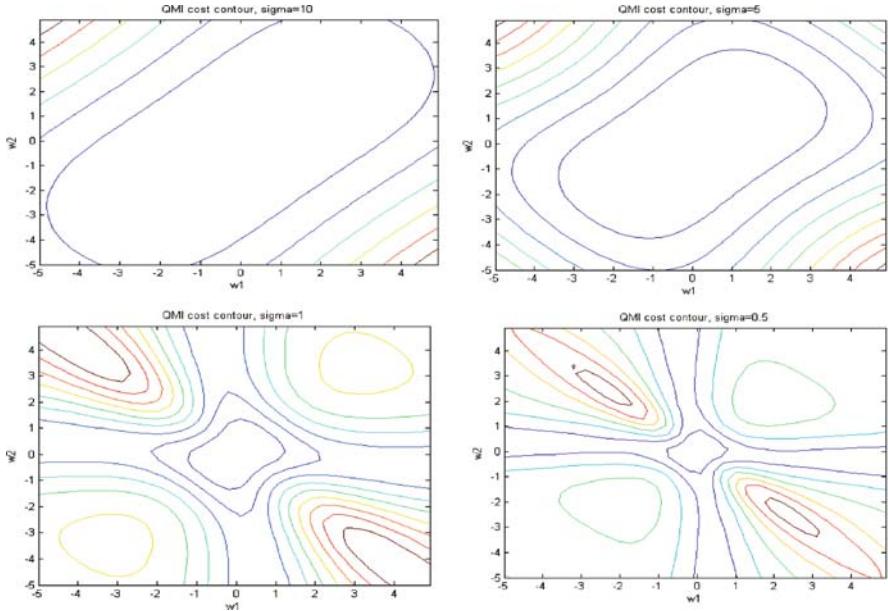


**Fig. 6.10.** Left panel: input to the two nonlinear PEs. Middle panel: output of the two nonlinear PEs showing the saturation that corresponds to the maximum QMI solution for this problem and system. Right panel: The final solution obtained in the output layer, which just flips one signal and adds the other.

starts at a large value ( $\sigma = 10$ ) and is annealed through training to  $\sigma = 0.5$  which is appropriate for the statistics of the sinewave with unit amplitude, and it is hoped, avoids the local minima of the performance surface. We select a batch size at least as large as one period of the input to provide a good description of the input PDF. Another slight modification is to use just the sign of the gradient (instead of the actual value) as suggested in Chapter 5 in the resilient backpropagation, because it makes training less dependent upon the actual structure of the performance surface (the existence of plateaus is one of the difficulties of neural network training). The stepsize is 0.01, and the number of training epochs is heuristically selected at 200.

Figure 6.10 shows from left to right the inputs to the nonlinear PEs and the corresponding hidden PE outputs that are trained first with  $\text{QMI}_{\text{ED}}$ . The mapping is very similar to the one found by conventional backpropagation learning [253]. Basically the solution requires that the hidden PEs saturate opposite sides of the input signal, and the output layer flips one and adds them as shown in the rightmost panel of the figure. We can now state that this solution also maximizes the mutual information at every layer of the deep network. It is particularly surprising that the saturation produced in the hidden layer, which is critical to build the double frequency, is a maximum mutual information solution for this problem with this type of nonlinearity.

Training the TDNN with  $\text{QMI}_{\text{ED}}$  is not simple, due to the local minima and shallow slopes. Figure 6.11 shows the effect of the kernel size in the performance surface to adapt the output weights and demonstrates the advantage of the kernel annealing. The two top panels obtained with a large kernel size show basically a convex performance surface but with a very flat and broad convexity, and the bottom panels, obtained with a smaller kernel size compatible with the data variance, show a well-defined global minimum but with many other local minima. Kernel annealing therefore helps avoid local minima but the current state solution still needs to be close to the optimum (hence the advantage of working just with the sign of the gradient).

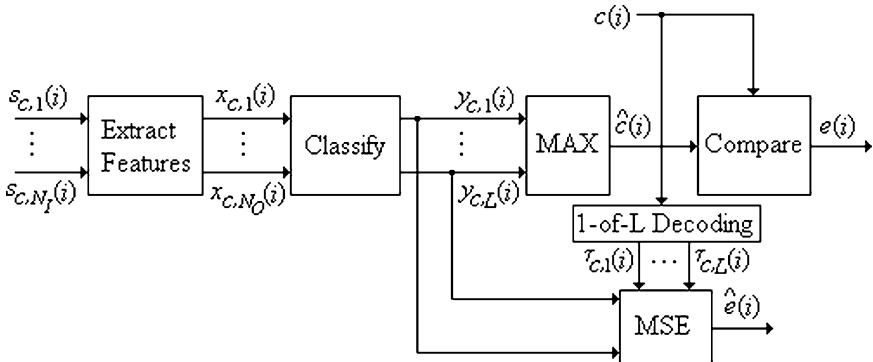


**Fig. 6.11.** The performance surface for the output layer obtained with different kernel sizes  $\sigma = 10, 5, 1, 0.5$  (clockwise from top left). Note the smoothing effect: for large kernel sizes, the solution is basically convex but with a very broad minimum. When the kernel size is reduced, the basin where the minimum exists is reduced, but other minima pop out, making the search much harder if the kernel is not annealed.

The QMI<sub>ED</sub> solution also has a few drawbacks with respect to the conventional training of the MLP with backpropagation. In fact, the cost function evaluates statistics of the input and desired responses, not matching sample-by-sample differences. Moreover, it works with pairs of samples of each signal so it is insensitive to the dc value, polarity, and to a certain extent also amplitude of the desired signal. Only the “PDF shapes” matter, which may be a hindrance or a great advantage, depending upon the application. So we are not yet ready to give up on backpropagation to adapt deep networks, but the proof of concept of adaptation with maximizing mutual information is established.

## 6.7 The Role of ITL Feature Extraction in Classification

In the previous section we showed how to use QMI to find an information theoretic linear subspace to represent data for classification (it can be a manifold if the projector is nonlinear). However, we loose the ability to determine the importance of each input as is required in feature selection. Here we compare ITL with more traditional methods of selecting and ranking individual



**Fig. 6.12.** Block diagram of a generic classification system (from [151]).

features, and with the design of optimal classifiers. We also introduce a more computationally efficient algorithm to train the projector.

The feature extractor and the classifier are shown in Figure 6.12, where  $s_c(i)$ ,  $\mathbf{x}_c(i)$ , and  $\mathbf{y}_c(i)$  are the inputs (size  $N_I \times 1$ ), the output features (size  $N_O \times 1$ ), and classifier outputs (size  $N_C \times 1$ ), respectively, for the  $i$ th exemplar, which happens to belong to class  $c$  ( $s(i)$ ) and  $\mathbf{x}(i)$  are used to denote the  $i$ th exemplar irrespective of class). Likewise  $c(n)$  and  $e(n)$  denote, respectively, the true class label and the error for the  $n$ th exemplar. Only linear transformations are considered here where  $\mathbf{x}_c(i) = \mathbf{R} s_c(i)$ . The feature extractor and the classifier can be trained independently as exemplified above, or simultaneously as used in several recent approaches. One important question is which method provides the best results.

Some researchers believe that training both systems at once, which involves the minimization of criteria that resemble the misclassification rate, are expected to outperform methods that train the selector and classifier independently. Herein, we use an information-theoretic method that trains the extractor in an independent fashion, and show that it outperforms several simultaneously trained systems on five randomly chosen datasets.

### Information-Theoretic Feature Extraction with Renyi's Entropy

The direct maximization of QMI as done for the SAR ATR example does not scale up well to many classes. Indeed, if there are many classes the QMI involves a joint with as many dimensions as classifier outputs, so estimating QMI with an acceptable error is computationally complex and demands lots of data that may not exist in a practical setting. There are, however, many other ways to write mutual information (MI), as we saw in Chapter 1. Shannon MI may be written in one of three equivalent expressions,

$$\begin{aligned}
I(\mathbf{X}, C) &= H(\mathbf{X}) - H(\mathbf{X} | C) \\
&= H(C) - H(C | \mathbf{X}), \\
&= H(\mathbf{X}) + H(C) - H(\mathbf{X}, C)
\end{aligned} \tag{6.29}$$

where  $H(\mathbf{X})$  is Shannon's entropy,  $\mathbf{X}$  is the random output feature vector, and  $C$  the class labels (where  $N$  realizations of  $\mathbf{X}$  and  $C$  are given by  $x(i)$  and  $c(i)$ , respectively, for  $i = 1, \dots, N$ ). Notice that the two first expressions do not need estimation in the joint space, which simplifies the computation and decreases the estimation error for a given number of data samples available. However, none of these expressions is commonly used because Shannon's entropy estimation is computationally intensive [21].

The method presented here and called maximization of Renyi's mutual information (MRMI) [151] replaces the two (Shannon) entropy terms in the first version of Eq. (6.29) with Renyi's entropy terms. If you recall the definition of Renyi's mutual information in Eq. (2.81), the change of the placement of the logarithm will not yield an equation similar to Eq. (6.29). However, multiplying the denominator by  $p_{X_o}(x_o)$  yields

$$I_\alpha(\mathbf{X}) \approx \frac{1}{\alpha - 1} \log \frac{\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} p_X^\alpha(\mathbf{x}) dx_1, \dots, dx_d}{\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \prod_{o=1}^n p_{X_o}^\alpha(x_o) dx_1, \dots, x_d} = \sum_{o=1}^d H_{R_\alpha}(x_o) - H_{R_\alpha}(\mathbf{x}) \tag{6.30}$$

which is a new figure of merit that maintains Shannon's general MI form but where every term is given by Renyi's entropy (however, it is no longer mutual information). For the case of two variables and using the first form of Eq. (6.29) yields,

$$I_2(\mathbf{X}, C) \approx H_2(\mathbf{X}) - H_2(\mathbf{X} | C), \tag{6.31}$$

where  $H_2(\mathbf{X})$  is Renyi's quadratic entropy. In spite of the approximation, this substitution is appealing because as we have seen in Chapter 4, Renyi's quadratic entropy accepts a stochastic approximator of the information potential, the SIG which was obtained by dropping the expected value Eq. (4.15) to reduce the computational complexity from  $O(N^2)$  to  $O(N)$ . Here, we use a Gaussian kernel with symmetric covariance matrix.

We selected Eq. (6.31) for the following reasons: the second formulation in Eq. (6.29) is not convenient for the entropy estimator with IP because the given data  $\mathbf{X}$  have a continuous distribution, thus requiring integration. The third form is used for blind source separation (BSS) [149] because the joint entropy can easily be made invariant to the adaptation, as we show in Chapter 8; however, this simplification does not apply when extracting features because the MI is measured between the output feature set and the class label. The entropy estimator with IP has a scale (gain) that is a function of the dimensionality of the underlying random vector [223]. This gain is irrelevant for the maximization or minimization of entropy when there is a single entropy

term, but it plays an important role when the criterion consists of a summation of two or more entropies if the random vectors on which they are based have different dimensionalities. To avoid the dimensionality-dependent gain for feature extraction, the first formulation in Eq. (6.29) is preferred because the two entropy terms have an identical dimensionality of  $N_O$ .

### Maximization of Renyi's Mutual Information (MRMI-SIG)

Both classifiers considered herein are made invariant to an invertible linear transformation to reduce the number of free parameters that must be adapted without unnecessarily restricting the possible set of decision surfaces that can be produced by a (linear) projection. This is achieved by constraining the feature extraction matrix  $\mathbf{R}$  to be a pure rotation matrix. Hence,  $\mathbf{R}$  can be expressed as a function of the  $N_O(N_I - N_O) \times 1$  vector of rotation angles,  $\theta$ , as follows.

$$\mathbf{x}_c(i) = \mathbf{R}(\theta)\mathbf{s}_c(i), \quad \mathbf{R}(\theta) = \left[ \prod_{k=1}^{N_O} \prod_{m=N_O+1}^{N_I} \mathbf{R}_{k,m}(\theta_{i,m}) \right]_{N_O}, \quad (6.32)$$

where the notation  $[\mathbf{A}]_{N_O}$  corresponds to keeping only the first  $N_O$  rows of matrix  $\mathbf{A}$ ,  $\theta_{k,m}$  is a single element of the rotation angle vector, and  $\mathbf{R}_{k,m}(\theta_{k,m})$  is an individual Given's rotation matrix [151]. Constraining the transformation in this manner reduces the number of parameters from  $N_O N_I$  to  $N_O(N_I - N_O)$ . The MRMI-SIG criterion can be estimated by substituting the SIG,  $H(\mathbf{x}) \cong -\log 1/N \sum_{i=1}^N G_{\sigma I}(\mathbf{x}(i) - \mathbf{x}(i-1))$  in Eq. (6.31) to yield

$$J = -\log \frac{1}{N} \sum_{i=1}^N G_{\sigma I}(\mathbf{x}(i) - \mathbf{x}(i-1)) + \sum_{c=1}^L \left( \frac{N_c}{N} \log \frac{1}{N_c} \sum_{i=1}^N G_{\sigma I}(\mathbf{x}_c(i) - \mathbf{x}_c(i-1)) \right), \quad (6.33)$$

where  $\mathbf{x}_c(i) = \mathbf{R}(\theta)\mathbf{s}_c(i)$  is the  $i$ th exemplar, which happens to belong to class  $c$ ,  $\Theta$  is the vector of rotation angles adapted to maximize  $J$ ,  $N_c$  is the number of class labels in the training set having class  $c$ , and  $N$  is the length of the training set. The parameters are updated using gradient ascent optimization, as seen in Chapter 4

$$\theta(i+1) = \theta(i) + \eta \nabla_{\theta} J_i,$$

where  $\eta$  is the step size and the subscript on  $J$  is used to denote that the order in which the data are presented is shuffled every iteration, the need for which is explained next.

The second term on the right-hand side of Eq. (6.33) increases when minimizing  $(\mathbf{x}_c(i) - \mathbf{x}_c(i-1))^2$ , which is accomplished by choosing  $\mathbf{R}$  such that all the consecutive exemplars from a given class are as near as possible to

each other in the space of the output features. This equates to minimizing the within-class spread in the limit as long as the data order is shuffled during adaptation. A trivial solution for minimizing the total within-class spread is to set  $\mathbf{R}$  equal to an all-zeros matrix. This, however, causes the features from all classes to overlap perfectly. The first term on the right-hand side of Eq. (6.33) prevents this undesirable solution because it is maximized by maximizing  $(\mathbf{x}(i) - \mathbf{x}(i-1))^2$ , which is a measure of the spread of the data (in the space of the output features) irrespective of the class. Linear discriminant analysis (LDA) is an alternate ways to construct a criterion that attempts to minimize within-class spread and maximize overall spread, [80] (which is based on second-order statistics). QMI and MRMI-SIG, which have both an information-theoretic interpretation, represent another possibility.

## Comparing Training Methodologies

The performances of several different methods are compared using the rate of correct classification of five different datasets. The two Bayes classifiers that are selected are the Bayes-P (parametric) and the Bayes-NP (nonparametric) classifiers, both of which generate nonlinear decision surfaces. The Bayes-P classifier assumes that the set of output features, for each class  $c$ , is a multivariate Gaussian distribution [108]. The Bayes-NP uses Parzen windows to estimate each of the a posteriori distributions as was done in Section 6.5. Unlike the Bayes-P classifier, the Bayes-NP classifier makes no assumptions on the distribution of the output features so that it is able to take into account higher-order statistics of the output features, including multiple modality. The Bayes-NP classifier uses a kernel size of  $\sigma = 0.25$ .

Results are shown for a total of six methods. Three of these train the extractor and classifier independently, namely, MRMI-SIG, PCA, and QMIED. The remaining three methods train the feature extractor and the classifier simultaneously. These methods include minimum classification error (MCE) [37], mean square error (MSE), and a method that ranks features based on classification performance of a validation set (FR-V) [38].

For the sake of perspective, the classification results of random projections are also included for the lower-dimensional datasets, the coefficients of which are chosen uniformly in  $[-1, 1]$ . The results of the random projection are represented in the plots using a dashed line. The MRMI-SIG, MCE, and MSE methods all have computational complexity  $O(N)$  and QMIED has computational complexity  $O(N^3)$ , whereas the computational complexity of FR-V depends only on the classifier and PCA has an analytical solution. For the two high-dimensional datasets, MRMI-SIG is used only to rank the input features so that the comparison between MRMI-SIG and FR-V is between two methods having similar computational complexity.

Feature ranking, which is suitable for datasets having extremely high dimensionality, is used for demonstrative purposes only. We suspect that using PCA to reduce the dimensionality to a manageable intermediate value or using

**Table 6.5.** Description of the Data Sets Used in the Comparison

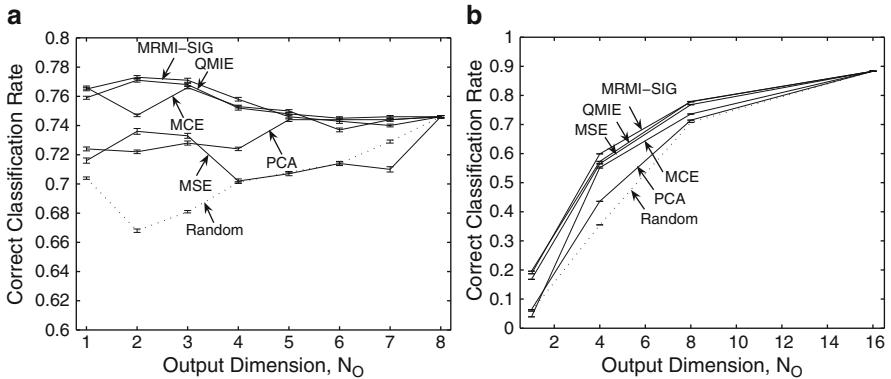
Dataset	$N_I$	$N_C$	$N_T$	Test Size	Outliers
Pima Indians	8	2	500	268	8%
Landsat Satellite Image (Statlog)	36	6	4435	2000	0%
Letter Recognition	16	26	16000	4000	0%
Musk	166	2	300	176	0%
Arrhythmia	279	16	300	152	0.3%

a multistage (semi greedy) approach will provide better classification results than ranking. MRMI-SIG uses a kernel size of  $\sigma = 0.5$  for all datasets (which is half of the data variance).

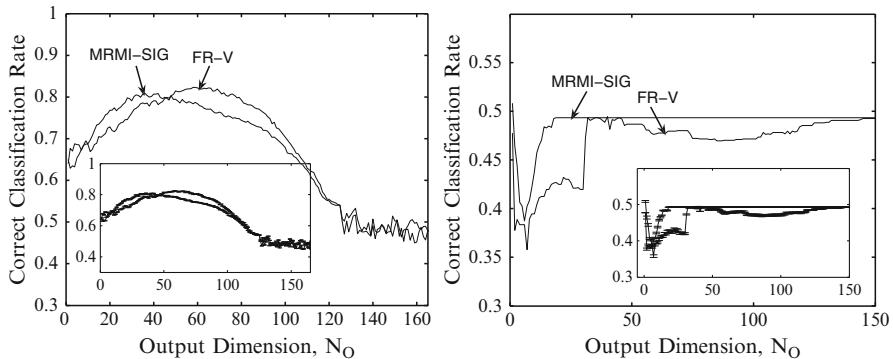
The MSE criterion uses the 1 out of the  $L$  scheme to define the targets,  $\tau_c(i)$ , which is defined as setting the target for the  $i$ th exemplar associated with class  $c$  to 1 and the other  $L - 1$  targets to 0 (this is represented in Figure 6.12 using the demultiplexer). MCE has two user-defined parameters,  $\alpha$  and  $v$ , which are set to 10 and 2, respectively. The FR-V method uses a validation set that is found by randomly selecting a (disjoint) subset of the original training set.

Table 6.5 shows the important characteristics of the five datasets. The first three were randomly selected from the list of all datasets at the UCI Machine Learning Repository, whereas the Musk and Arrhythmia datasets were selected for their large input dimensionality (all data sets may be found at <http://www.ics.uci.edu/~mlearn/MLRepository.html>). For all methods except PCA, data were preprocessed such that the resulting ( $N_I \times 1$ ) input features,  $s(i)$ , are zero-mean, (spatially) uncorrelated, and have unit variance. Because the transform for PCA depends on the eigenvalues of the autocorrelation matrix, spherling should not be used with PCA. The Pima dataset has numerous invalid datapoints, for example, features that have a value of 0 even though a value of 0 is not meaningful or physically possible. These correspond to points in feature space that are statistically distant from the mean calculated using the remaining data (with the points in question removed). No attempt was made to remove or correct for these outliers. Likewise, the Arrhythmia dataset has missing values, all of which are set to 0 for the comparison.

The Bayes-P classifier produces the best classification performance for the Pima, Landsat, and Musk datasets, whereas the Bayes-NP classifier performs best for the Letter Recognition and Arrhythmia datasets. Therefore, the results shown are restricted to these combinations of datasets and classifiers. This choice has no effect on the relative performance of the different feature extraction methods. The training is performed using  $N_T$  randomly selected samples of the dataset and is tested on the remaining (disjoint) data. All results are reported using ten-fold cross-validation. The results for all algorithms are the same whenever  $N_O = N_I$ . This is because both classifiers are invariant under full-rank linear transformations.



**Fig. 6.13.** Classification performance versus output dimension for PIMA (a) and Letter data (b). (from [151]).



**Fig. 6.14.** Classification performance versus output dimension for Musk and Arrhythmia data (from [151]).

Figure 6.13 shows the correct classification rates for the Pima and Letter Recognition data. Each figure includes error bars that represent one standard error. All five methods perform well on these datasets, with the exception that MSE has trouble with the Pima dataset and PCA performs poorly for the Letter Recognition data for small  $N_O$ . The proposed method has the best performance for the Pima and Letter Recognition datasets. It is interesting to recognize that the differences in the methods are most noticeable for the small projection spaces, and when  $N_O$  increases, even the random projections have similar performance; that is, there is progressively less advantage in fine tuning the projector for larger projection spaces.

Figure 6.14 shows the results for the two high-dimensional datasets, Musk and Arrhythmia. For these two plots, MRMPI-SIG is only used to rank the features. Results are shown for both MRMPI-SIG and FR-V and the inset in each figure shows the corresponding error bars. In Figure 6.14a, the best

performances for MRMI-SIG and FR-V are similar. Notice, however, that MRMI-SIG concentrates the most useful group of features for classification into the top 20% of the highest-ranked features.

Hence, it requires roughly 25 fewer features to reach peak performance. The jaggedness of the curves for  $N_O > 120$  corresponds to the point at which at least one of the class covariance matrices used in the Bayes-P classifier becomes ill-conditioned. The curves in Figure 6.14b have a very different characteristic due to the use of the Bayes-NP classifier where roughly 1/4 of the 16 class labels are poorly represented (fewer than five instances each).

Classification results for both methods are flat and very nearly equal for the portion of  $N_O$  not shown in the figure. As can be seen, the performance of MRMI-SIG is better than or equal to the performance of FR-V for all values of  $N_O$ .

These results should be contrasted with the formulation of the MCE method and the view expressed by LeCun et al. [195] that discriminative (simultaneously trained) methods should be the preferred way to train classifiers. Although the arguments make sense, we have not found any practical advantage for discriminative training in this study, and conclude that the issue may be problem-dependent [151]. In fact discriminative training and training one class at a time puts emphasis on different parts of the data space (boundaries between classes and cluster centers, respectively), so the nature of the problem has an important role in the final results.

## 6.8 Error Bounds for Classification

Fano's bound [97] is a well-known inequality because it is essential to prove key theorems in information theory [65]. Fano's bound entails an important conclusion about the structure of optimal classifiers: the probability of error is lower bounded by the mutual information between data classes. But it does so *a priori*, just by examining the structure of the data without implementing any classifier.

The related question of determining optimal features has been one of the major focal points in pattern recognition research, and information theory has also played a central role in this quest. It has been established that information is not preserved in subspace projections, yet information maximization across the mapping is essential in this process [68]. As we saw in this chapter, mutual information can be used to train classifiers directly and is also useful for feature extraction using the nonparametric estimator for quadratic Renyi's entropy. Although Fano's lower bound for the probability of error in classification is a valuable indicator of attainable performance, the goal in statistical pattern recognition and machine learning is to minimize the probability of error [268], or possibly an upper bound for the error probability as in structural risk minimization [323]. Therefore, a family of lower and upper bounds would encompass the advantages of both: identify the limitations and indicate the

possible generalization performance simultaneously. In the theory of bounds, one of the central issues is to study how tight the bound is because otherwise its usefulness diminishes.

The flexibility of Renyi's family of entropy and divergences is also central to our results because it encompasses Shannon's definitions as special cases. The recent work of Feder and Merhav is especially important as it provides a lower and an upper bound for the minimal probability of error in estimating the value of a discrete random variable [98]. Their bounds show the association between the probability of value-prediction error and Shannon's entropy, and Renyi's entropy with infinite order as well. Han and Verdu's generalization of Fano's bound, again using  $\alpha = \infty$  Renyi's entropy is theoretically appealing and also useful in proving a generalized source-channel separation theorem [132]. Yet, the bounds presented in these works do not explicitly consider the classification process, thus do not make use of the confusion matrix of the classifier under consideration. We develop here a family of lower and upper bounds, using Renyi's definitions of information-theoretic quantities [86]. The free parameter  $\alpha$  in Renyi's definitions was exploited along with Jensen's inequality for convex and concave functions.

In order to understand the effect of  $\alpha$  on the value of entropy, consider the following fact: Renyi's entropy is a monotonically decreasing function of  $\alpha$  whose values range from  $\log N_c$  to  $-\log(\max_k p(c_k))$ , where  $N_c$  is the total number of events and  $p(c)$  is the probability of each event as  $\alpha$  is varied from zero to infinity. As shown in Chapter 2 the limit of Renyi's entropy (and mutual information) approaches Shannon's definitions as  $\alpha$  goes to 1.

### Fano's Bound on Misclassification Probability

Fano's inequality determines a lower bound for the probability of classification error in terms of the information transferred through the classifier. More specifically, consider a classifier for which the actual classes ( $N_c$  is the number of total classes), denoted by  $C$ , have prior probabilities  $\{p(c_k)\}_{k=1}^{N_c}$  and the classifier decisions, denoted by  $Y$ , have the conditional probabilities  $p(y_j|c_k)$  where the index  $j$  indicates the classifier decision. Fano's bound for the probability of classification error, in terms of the conditional entropy, is given by [97]

$$p_e \geq \frac{H_S(Y|C) - h_S(p_e)}{\log(N_c - 1)} = \frac{H_S(Y) - I(Y, C) - h_S(p_e)}{\log(N_c - 1)}, \quad (6.34)$$

where

$$H_S(Y|C) = \sum_{k=1}^{N_c} H_S(Y|c_k)p(c_k) \quad \text{and}$$

$$H_S(Y|c_k) = -\sum_{j=1}^{N_c} p(y_j|c_k) \log p(y_j|c_k).$$

The special notation  $h_S(p_e) = -p_e \log p_e - (1 - p_e) \log(1 - p_e)$  is Shannon's entropy of the binary error. Notice that Eq. (6.34), as it appears in Fano's derivation has the probability of error appearing on both sides of the inequality. Also the denominator prevents the application of this bound to two-class situations. To account for these problems, the binary entropy of  $p_e$  is replaced by its maximum possible value,  $\log_2 2 = 1$ , and the denominator is replaced with the larger  $\log N_c$ . In addition, the conditional entropy can be replaced by the sum of marginal entropy and mutual information terms in accordance with Eq. (6.29), yielding the commonly presented version of Fano's bound in the literature [318]

$$p_e \geq \frac{H_S(Y) - I_S(Y, C) - 1}{\log N_c}. \quad (6.35)$$

Notice, however, that substituting  $h_S(p_e)$  by a reasonable quantity without implementing a classifier is not a trivial problem and it is key for the tightness of the bound. This issue is not pursued here (see [86]).

### Bounds Using Renyi's Entropy and Mutual Information

We apply Jensen's inequality on Renyi's definition of conditional entropy, joint entropy, and mutual information to obtain the following lower and upper bounds for the probability of error [86]. Renyi's mutual information and conditional entropy do not share the identity in Eq. (6.34), thus these bounds have to be separately derived, starting from their corresponding basic definitions. For simplicity, we only provide the derivation for the bound that uses the conditional entropy below. The derivations of the bounds using the joint entropy and the mutual information are given in [92].

*Jensen's Inequality:* Assume that  $g(x)$  is a convex function (if concave reverse inequality), and  $x \in [a, b]$ ; then for  $\sum_k w_k = 1$ ,  $w_k > 0$ ,  $g(\sum_k w_k x_k) \leq \sum_k w_k g(x_k)$ .

For later use in the derivation, we also write the conditional probability of error given a specific input class  $k$  as

$$p(e | c_k) = \sum_{j \neq k} p(y_j | c_k) - 1 - p(e | c_k) = p(y_k | c_k). \quad (6.36)$$

Consider Renyi's conditional entropy of  $Y$  given  $c_k$ .

$$\begin{aligned} H_\alpha(Y | c_k) &= \frac{1}{1-\alpha} \log \sum_j p^\alpha(y_j | c_k) = \frac{1}{1-\alpha} \log \left[ \sum_{j \neq k} p^\alpha(y_j | c_k) + p^\alpha(y_k | c_k) \right] \\ &= \frac{1}{1-\alpha} \log \left[ p^\alpha(e | c_k) \sum_{j \neq k} \left( \frac{p(y_j | c_k)}{p(e | c_k)} \right)^\alpha + (1 - p(e | c_k))^\alpha \right] \end{aligned} \quad (6.37)$$

Using Jensen's inequality in Eq. (6.37), and the fact that  $H_S(e|c_k) = p(e|c_k) \log p(e|c_k) + (1 - p(e|c_k)) \log(1 - p(e|c_k))$ , we obtain two inequalities for  $\alpha > 1$  and  $\alpha < 1$  cases:

$$\begin{aligned} H_\alpha(Y|c_k) &\stackrel{\alpha>1}{\leq} p(e|c_k) \frac{1}{1-\alpha} \log p^{\alpha-1}(e|c_k) \sum_{j \neq k} \left( \frac{p(y_j|c_k)}{p(e|c_k)} \right)^\alpha \\ &\quad + (1 - p(e|c_k)) \frac{1}{1-\alpha} \log(1 - p(e|c_k))^{\alpha-1}. \quad (6.38) \\ &\stackrel{\alpha<1}{\geq} H_S(e|c_k) + p(e|c_k) \frac{1}{1-\alpha} \log \sum_{j \neq k} \left( \frac{p(y_j|c_k)}{p(e|c_k)} \right)^\alpha \end{aligned}$$

Let us call  $H_\alpha(Y|e, c_k)$  the conditional entropy given an error is made in classification and the actual class was  $c_k$ , which is computed as

$$H_\alpha(Y|e, c_k) = \frac{1}{1-\alpha} \log \sum_{j \neq k} \left( \frac{p(y_j|c_k)}{p(e|c_k)} \right)^\alpha. \quad (6.39)$$

We now have to find an upper bound and a lower bound for this quantity depending upon the value of  $\alpha$ . Recall that for an  $(N_c - 1)$  sample set, we have the following upper bound for entropy, which is the entropy of a uniform probability distribution,

$$\frac{1}{1-\alpha} \log \sum_{j \neq k} \left( \frac{p(y_j|c_k)}{p(e|c_k)} \right)^\alpha \leq \log(N_c - 1), \quad (6.40)$$

equality being achieved only for a uniform distribution. Hence, for  $\alpha > 1$ , from Eq. (6.38) and Eq. (6.40) we obtain

$$H_\alpha(Y|c_k) \leq H_S(e|c_k) + p(e|c_k) \log(N_c - 1). \quad (6.41)$$

Finally, using Bayes rule on the conditional distributions and entropies we get the lower bound for  $p_e$  as

$$H_\alpha(Y|C) \leq H_S(e) + p_e \log(N_c - 1) \rightarrow p_e \geq \frac{H_\alpha(Y|C) - H_S(e)}{\log(N_c - 1)}. \quad (6.42)$$

For  $\alpha < 1$ , from Eq. (6.38) we have

$$\begin{aligned} H_\alpha(Y|c_k) &\geq H_S(e|c_k) + p(e|c_k) H_\alpha(Y|e, c_k) \\ &\geq H_S(e|c_k) + p(e|c_k) [\min_k H_\alpha(Y|e, c_k)]. \end{aligned} \quad (6.43)$$

Finally, combining these results and incorporating Fano's special case into the lower bound, we bracket the classification error probability as,

$$\frac{H_\alpha(Y|C) - H_S(e)}{\log(N_c - 1)} \leq p_e \leq \frac{H_\beta(Y|C) - H_S(e)}{\min_k H_\beta(Y|e, c_k)}, \quad \alpha \geq 1, \beta < 1. \quad (6.44)$$

Following a similar approach [92], we obtain the following upper and lower bounds for  $p_e$  expressed in terms of the joint entropy and the mutual information.

$$\frac{H_\alpha(Y, C) - H_S(C) - H_S(e)}{\log(N_c - 1)} \leq p_e \leq \frac{H_\beta(Y, C) - H_S(C) - H_S(e)}{\min_k H_\beta(Y|e, c_k)}, \quad \begin{aligned} \alpha &\geq 1 \\ \beta &< 1 \end{aligned} \quad (6.45)$$

$$\frac{H_S(Y) - I_\alpha(Y, C) - H_S(e)}{\log(N_c - 1)} \leq p_e \leq \frac{H_S(Y) - I_\beta(Y, C) - H_S(e)}{\min_k H_\beta(Y|e, c_k)}, \quad \begin{aligned} \alpha &\geq 1 \\ \beta &< 1 \end{aligned} \quad (6.46)$$

Notice that in all three cases, the lower bounds for  $\alpha = 1$  correspond to Fano's bound through Eq. (6.35). The term in the denominator of the upper bound is the entropy of the conditional distribution given the actual class under the condition that the classifier makes an error.

From a theoretical point of view, these bounds are interesting as they indicate how the information transfer through the classifier relates to its performance. Because the family parameter of Renyi's definition does not affect the location of minimum and maximum points of the entropy and mutual information, it is safely concluded from Eq. (6.45) that, as the mutual information between the input and the output of the classifier is increased, its probability of error decreases. Consequently, this result also provides a theoretical basis for utilizing mutual information for training classifiers and performing feature extraction.

The denominators of the upper bounds also offer an interesting insight about the success of the classification process. As these entropy terms are maximized, the upper bounds become tighter. This happens when the corresponding distribution is uniform; that is, when the distribution of probabilities over the erroneous classes is uniform. This conforms to the observations of Feder and Merhav [98] who noted that in a prediction process, their upper bound is tightest when the probabilities are distributed uniformly over the wrong values.

Recall that Renyi's entropy is a monotonically decreasing function of the entropy order. Therefore, it is clear that the lower bound in Eq. (6.44) attains its tightest (i.e., greatest) value for Shannon's entropy, which is exactly Fano's bound. Determining the tightest upper bound is not as easy. The optimal value of the entropy order is determined by the balance between the decrease in the numerator and the increase in the denominator. However, our simulations with several simple examples point out that the tightest value for the upper bounds may as well be attained for values of entropy order approaching 1 from above. These simulation results are presented below.

One issue to be solved in these bound expressions (also an issue for the original bound by Fano) is to eliminate the binary entropy of the probability of error from the bounds; otherwise, the probability of error appears in both

sides of the inequalities. For theoretical use, this does not cause any problems (as evident from the wide use of Fano's bound in various proofs in information theory). From a practical point of view, however, this situation must be corrected. We investigated ways of achieving this objective [86], however, the obtained bounds were extremely loose compared to the original bounds. Therefore, we do not present these modified bounds here. We could use the bounds as they appear in Eqs. (6.44) to (6.46) by nonparametrically estimating the confusion matrix and the prior probabilities (perhaps by simply counting samples). On the other hand, the information used in this approach is already sufficient to estimate directly the probability of error itself. Therefore, we suggest using the estimated bounds as a confirmation of the estimated probability of error. They may also provide confidence intervals on the calculated value. For a practical application of this procedure, however, further work and analysis of the bounds estimated from a finite number of samples is necessary [86].

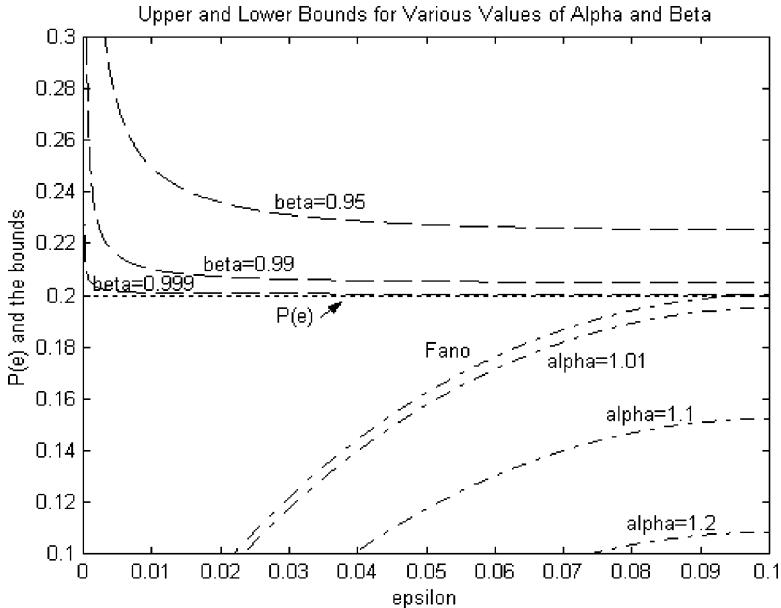
## Numerical Illustrations of the Bounds

In this section, we show the performance of the bounds in two different numerical case studies. These studies are aimed at showing the basic conclusions drawn in the preceding sections about these Renyi bounds. In addition, we present a comparison of these bounds and the Feder and Merhav bound applied to misclassification probability through one of the examples. Our first example is a simple three-class situation designed to test the basic properties of the bounds. For this example, the confusion matrix of our hypothetical classifier is given by

$$P_{Y|C} = \begin{bmatrix} 1 - p_e & p_e - \varepsilon & \varepsilon \\ \varepsilon & 1 - p_e & p_e - \varepsilon \\ p_e - \varepsilon & \varepsilon & 1 - p_e \end{bmatrix},$$

whose  $i, j$ th entry denotes the conditional probability of decision on class  $i$  given the input class  $j$ . Each column represents the distribution of the probabilities among the possible output classes and the diagonal entries correspond to the probabilities of correct classification given a specific input class. The structure of this confusion matrix guarantees that the overall probability of error is fixed at  $p_e$ , which is selected to be 0.2 in the following examples. By varying the free variable  $\varepsilon$  in the interval  $[0, p_e/2]$ , it is possible to study the performance of the bounds in terms of tightness. The lower and upper bounds of Eq. (6.44) evaluated for various values of the family parameter (entropy order) are shown in Figure 6.15 as a function of  $\varepsilon$ .

We observe that the family of lower bounds achieves its tightest value for Fano's bound, whereas the upper bounds become tighter as the entropy order approaches one. One other interesting observation is that the upper bounds remain virtually flat over a wide range of  $\varepsilon$ , suggesting that this bound is as



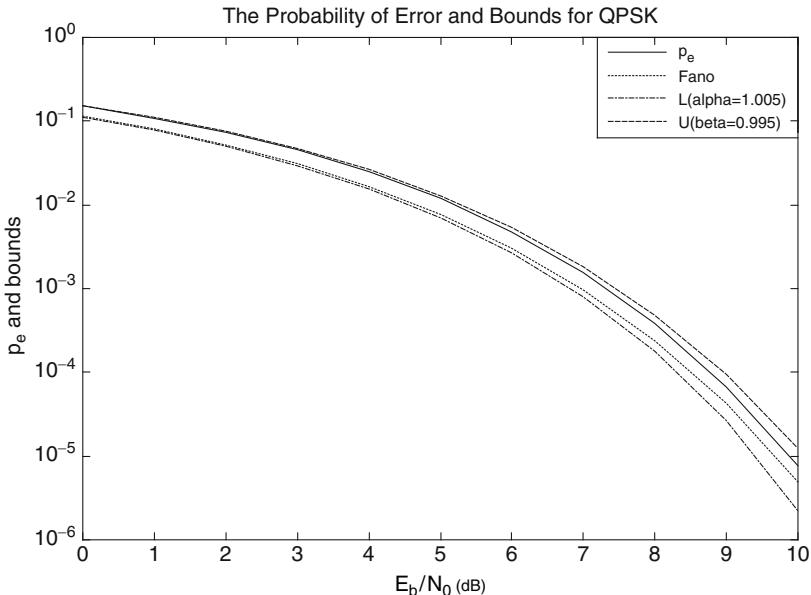
**Fig. 6.15.** Family of lower and upper bounds for probability of error evaluated for different values of Renyi's entropy order (from [92]).

tight for a broad variety of classifiers as it is for the optimum situation where the probability mass distribution among the wrong output classes is uniform. If the upper bound is evaluated for  $\beta = 1$ , then it reduces to exactly the probability of error,  $p_e$ .

*QPSK Example.* As a second example, we evaluate the bounds for an oversimplified quadrature phase-shift keying (QPSK) digital communication scheme over an additive white Gaussian noise (AWGN) channel. The energy per transmitted bit is  $E_b$  and the power spectral density (PSD) for the additive white Gaussian noise is  $N_0/2$ . In this problem, it is possible to evaluate the exact values for average bit error rate  $p_e$  and all the conditional and prior probabilities necessary to evaluate the bounds in terms of  $Q$ -functions. The confusion matrix for this case is

$$P_{Y|C}^{QPSK} = \begin{bmatrix} (1 - Q_1)^2 & Q_1^*(1 - Q_1) & Q_1^2 & Q_1^*(1 - Q_1) \\ Q_1^*(1 - Q_1) & (1 - Q_1)^2 & Q_1^*(1 - Q_1) & Q_1^2 \\ Q_1^*(1 - Q_1) & Q_1^*(1 - Q_1) & (1 - Q_1)^2 & Q_1^*(1 - Q_1) \\ Q_1^2 & Q_1^2 & Q_1^*(1 - Q_1) & (1 - Q_1)^2 \end{bmatrix},$$

where  $Q_1 = Q\left(\sqrt{2E_b/N_0}\right)$ . The prior probabilities for each symbol are assumed to be uniformly distributed. The probability of error and the bounds are shown in Figure 6.16. For the upper bound, we used entropy order 0.995 in Eq. (6.44).



**Fig. 6.16.** Probability of error and its bounds versus bit-energy-to-noise ratio for QPSK (from [92]).

As we can see in these simple examples, the bounds do not differ much from the theoretical probability of error, but a full study of this issue is beyond the scope of this book.

## 6.9 Conclusions

This chapter addressed the design of classifiers with ITL cost functions. We started by extending the EEC to classification, but we found that the problem is more complex, because in classification the error is also conditioned on the class. From a theoretical perspective it is still unclear under what conditions minimizing the error entropy in classification is optimum. However, the experimental results with the MEE criterion yield results that supplant the ones obtained with MLPs trained with MSE.

In this chapter we derived the first algorithms to train linear and nonlinear systems with divergence or quadratic mutual information. These costs benefit from the IP estimator, and so it is just a matter of modifying the injected error to train the learning machine. We have applied these algorithms to train classifiers for automatic target recognition with very good results. We also applied mutual information to extract features for classification, and verified that the IP estimator is capable of extracting features that yield classification results comparable to or better than alternative methods.

Finally, we addressed the very important relationship between the probability of error and the mutual information proved by Fano. In order to improve our pattern recognition and feature extraction capabilities, it is imperative to understand how the information propagation through the classifiers and feature extractors affects the overall performance. Fano's bound provides the attainable limits for performance. However, the bounds we derived in this chapter provide upper bounds for the probability of classification error, which is an important result to evaluate the generalization capabilities. The key conclusion of this chapter is that, by training classifiers to maximize the mutual information between its output and decision classes, it is possible to decrease the probability of error, because both the lower and the upper bound decrease in this situation. Moreover, both the entropy of the right decisions and the entropy of the wrong decisions are important in determining the final performance.

These arguments should be contrasted with the MEE discussion of Section 6.2. They indicate that for classification, estimating the error entropy at the output of the classifier may be suboptimal. At the very least one should use discriminative training, which may explain why the information potential in batch mode works better in practice than the theoretical analysis predicts. But the intriguing idea for classification is to train non parametrically discriminative projections with mutual information.

## Clustering with ITL Principles

Robert Jenssen and Sudhir Rao

### 7.1 Introduction

Learning and adaptation deal with the quantification and exploitation of the input source “structure” as pointed out perhaps for the first time by Watanabe [330]. Although structure is a vague and difficult concept to quantify, structure fills the space with identifiable patterns that may be distinguishable macroscopically by the shape of the probability density function. Therefore, entropy and the concept of dissimilarity naturally form the foundations for unsupervised learning because they are descriptors of PDFs.

In the case of supervised learning there is a major simplification because labels are available and normally they are of the same dimensionality of the system output. Therefore one can define a composite random variable, the error that contains information about the differences in the distribution of the desired and the system output that simplifies the adaptation.

In unsupervised learning the researcher only has the input data (i.e., only one source of information). Finding structure in the data then requires a methodology that will be able to quantify in detail similarity and one or more criteria that somehow impose constraints to elucidate relationships among the data samples. We can expect that the evaluation of dissimilarity (or similarity), normally quantified by divergence becomes the center piece of unsupervised learning.

We have seen that the information potential (the expected value of the PDF), the core quantity of ITL algorithms, evaluates pairwise data interactions and extracts more information from data than evaluations based on single data samples such as the mean or even the variance (which computes distances between two samples, but one of them – the mean – is fixed, therefore it does not require full pairwise evaluations). Therefore, as explained in Chapter 2, the pairwise interactions of the ITL algorithms can be used naturally as entries of a similarity matrix when the goal of the analysis is mostly based on the estimation of information divergences. Moreover, because ITL provides an

efficient nonparametric estimator of entropy, it allows the description of data structure in a more meaningful way than what is normally done with variance.

In this chapter we present two basic procedures based on entropy and divergence to perform clustering, one of the centerpieces of unsupervised learning. The concept of IP and its corresponding forces is a powerful analogy for understanding information-theoretic clustering. A cost function based on the cross information potential is proposed and a gradient algorithm developed to optimize the cost. Examples and comparisons show that this criterion is able to cluster data of arbitrary shapes provided that there is a “valley” between them. We also present the relationship of ITL clustering to spectral clustering, graph cuts, and to mean shift algorithms which are faster algorithms based on a fixed-point update.

## 7.2 Information-Theoretic Clustering

Clustering [136] is a data grouping technique that obeys a given measure of similarity. Clustering algorithms attempt to organize unlabeled feature vectors into clusters or “natural groups” such that samples within a cluster are “more similar” to each other than to samples belonging to different clusters. In clustering there is neither information given about the underlying data structure nor is there a single similarity measure to differentiate all clusters. Hence, it should not come as a surprise that there is no unifying clustering theory.

The literature on clustering is huge [136]. Comprehensive introductory concepts can be found in the books by Duda et al [80], and Jain and Dubes [160]. Clustering algorithms can be basically divided into two groups depending upon locality or globality of the criterion. In the former case, some attribute of the local structure of the data is tested against the local criterion, in order to construct the clusters. This category includes hierarchical clustering algorithms [243], and algorithms based on artificial neural networks [253]. The majority of clustering metrics are based on a minimum variance criterion, for instance, merging and splitting, neighborhood-dependent, hierarchical methods and ART networks [115]. Competitive networks [183] can be used in clustering procedures where they form Voronoi cells with a minimum variance flavor. Valley seeking clustering [108] is a different concept that exploits not the regions of high sample density but the regions of less data. In a sense valley seeking clustering attempts to divide the data in a way similar to supervised classifiers, that is, by positioning discriminant functions in data space to provide nonlinear discriminants among the data clusters (unlike the previous methods). The problem in valley seeking clustering is the pulverization of the number of clusters, if there is a slight nonuniformity in one of them. Traditional valley seeking clustering still uses a variance measure to split the data.

The most popular global optimization technique for clustering is MacQueens’ k-means algorithm [208]. The idea in k-means is to find the

best division of  $N$  samples by  $k$  clusters  $C$  for  $i = 0, \dots, K$  such that the total distance between the clustered samples and their respective centers (i.e., the total variance) is minimized. The criterion as an equation reads

$$\min_x J = \sum_{i=1}^K \sum_{n \in C_i} |\mathbf{x}_n - \boldsymbol{\gamma}_i|^2, \quad (7.1)$$

where  $\boldsymbol{\gamma}_i$  is a vector representing the center of the  $i$ th class. This criterion is similar to local regression, except that in clustering, the residuals are the distance between each point and its cluster center. Minimizing the residuals (total variance) provides the “best” clustering. The k-means algorithm starts by randomly assigning samples to the classes  $C_i$ , computes the centers according to

$$\boldsymbol{\gamma}_i = \frac{1}{N_i} \sum_{n \in C_i} \mathbf{x}_n, \quad (7.2)$$

then reassigns the samples to the nearest cluster, and reiterates the computation. One can show that  $J$  decreases at each step until it reaches a minimum. If we interpret  $J$  as a performance criterion, its minimum can be obtained by taking the gradient with respect to the unknowns (the centers’ positions). With the new sample  $\mathbf{x}(n)$ , one first implements a competitive step by asking which is the cluster center  $\boldsymbol{\gamma}_i$  that is closest to the sample. Then use an on-line update algorithm that will move the center  $\boldsymbol{\gamma}_i$  incrementally towards the new sample according to

$$\boldsymbol{\gamma}_i(n+1) = \boldsymbol{\gamma}_i(n) + \eta(\mathbf{x}(n) - \boldsymbol{\gamma}_i(n)). \quad (7.3)$$

One can show that with this very simple procedure and the  $L_2$  metric, convex clusters (think of spheres) will be created. Note that the user has to select the number of clusters beforehand. There are many applications of clustering with great practical significance such as speech recognition, handwritten character classification, fault detection, and medical diagnosis. As can be expected the simplicity of k-means has a cost in performance. Therefore what we would like to investigate here is if the ITL ideas can improve performance through reclustering of the samples, i.e. when these simple minimum variance clusters are considered as an initial condition.

Information-theoretic methods appear particularly appealing alternatives to capture data structure beyond second order statistics. This stems from the fact that entropy and PDF distance measures are scalars that summarize in relevant ways the information contained in the data distributions. Previously, Watanabe [330] used a coalescence model and a cohesion method to aggregate and shrink the data into desired clusters. In recent years, Rose, et al [271] employed the robustness properties of maximum entropy inference for vector quantization, and Hofmann and Buhmann [153] applied the same criterion for pairwise clustering. Roberts et al. proposed a clustering method based on minimizing the partition entropy. Lately, Tishby and Slonim [317] proposed

the mutual information based information bottleneck method, and reported successful results. Information-theoretic clustering methods have also been put forward in the context of independent component analysis (ICA) [13].

The major problem of clustering based on information-theoretic measures has been the difficulty of evaluating the metric without imposing unrealistic assumptions about the data distributions, and this is the reason ITL is so promising. We start with the simplest of the clustering methods, k-means clustering [208], and apply the sample estimators of Renyi's entropy and quadratic divergences to the formed clusters to find out how they improve clustering. We claim [162] that there are at least two ways to apply ITL concepts to enhance performance through reclustering k-means assignments:

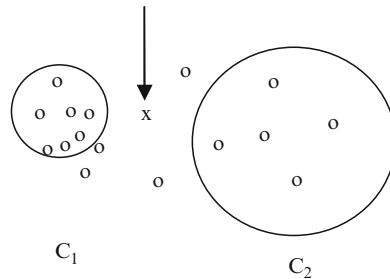
1. Assign the current sample to the cluster that minimizes the increase in differential entropy.
2. Assign the current sample to the cluster that maximizes the distance among the clusters.

Either method can be implemented with the ITL descriptors of information potential and Euclidean or Cauchy–Schwarz divergences covered in Chapter 2. Differential clustering is intuitive and is a straight application of the concepts of the information potential that we covered in Chapter 2. There are, however, practical problems with differential entropy clustering (entropy is not scale-invariant, so it is difficult to compare entropy of datasets with different numbers of samples and different cluster shapes) which make the method difficult to apply in practice [165]. The idea of maximizing divergences is much more robust, but it needs a more in-depth study. We present a cluster evaluation function that is central to clustering and is related to the cross-information potential (CIP) [114]. When first developed, both differential clustering and the concept of clustering evaluation function (CEF) were applied with greedy methods.

### 7.3 Differential Clustering Using Renyi's Entropy

From a theoretical viewpoint, clustering is a form of nonparametric density estimation. In the absence of a desired response, the best we can do for categorization is to use the information about the input data distribution to separate inputs into groups that share the same region in data space. The basic idea of clustering is to seek regions of high sample density – data clusters – and represent their centers. Cluster centers thus represent the local modes of the data distribution, where the definition of local is based on the number of centers we are willing to have.

Consider the situation depicted in Figure 7.1. A set of patterns, or feature vectors, is distributed in feature space. Initially a sufficiently large subset of the feature vectors has been assigned to clusters  $C_1$  or  $C_2$ . These are shown as the encircled points. The problem is now to decide whether a new pattern  $x$  (pointed to by the arrow) should be assigned to  $C_1$  or  $C_2$ .



**Fig. 7.1.** Assigning a new sample to pre-existing clusters.

We propose to cluster  $x$  based on the following simple observation. If  $x$  is wrongly assigned to  $C_1$ , the uncertainty, or entropy, of  $C_1$  will increase more than the entropy of  $C_2$  when  $x$  is added to it. Hence, in the general case of having initial clusters  $C_k$ ,  $k = 1, \dots, K$ , assign  $x$  to cluster  $C_i$  if

$$H(C_i + x) - H(C_i) < H(C_k + x) - H(C_k) \quad \text{for } k = 1, \dots, K \quad k \neq i, \quad (7.4)$$

and  $H(\cdot)$  denotes the entropy of cluster  $C_k$ . We refer to this method as *differential entropy clustering*. The estimation of entropy directly from data can be implemented with the information potential estimator of Renyi's quadratic entropy (Eq. 2.14). Because the entropy is calculated based on points assigned to the same cluster, we refer to Eq. (7.4) as the *within-cluster entropy*. From the proximity matrix  $D$  created by all pairwise distances, the similarity matrix with elements  $\{V_{ij} = \kappa(x_i - x_j)\}$  is computed and stored, where  $\kappa$  is the kernel used to estimate the IP. This means that after the similarity matrix has been obtained, all calculations are matrix manipulations. Obviously this limits somewhat the size of the dataset that can be clustered with this algorithm.

Another issue regards the kernel size  $\sigma$ . Our experiments have shown that promising clustering results are obtained provided that  $\sigma$  is chosen such that the Parzen PDF estimate is relatively accurate. It should be noted that this algorithm resembles other kernel-based clustering methods, such as spectral clustering [13] and Mercer kernel-based clustering [162]. In all such methods, as in ITL, the kernel size is a parameter of great importance defining the scale of the analysis.

## 7.4 The Clustering Evaluation Function

Let's assume that we have two data subgroupings  $p(x)$  and  $q(x)$ . As briefly mentioned in Chapter 2, the cross-information potential  $\int p(x)q(x)dx$  measures dissimilarity (i.e., some form of “distance”) between the two data subgroups. For quick reference we include below the definition of CIP assuming a Gaussian kernel,

$$\hat{V}(p, q) = \frac{1}{N_1 N_2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} G_{\sigma\sqrt{2}}(x_i - x_j) \quad (7.5)$$

with  $x_i \in p(x)$  and  $x_j \in q(x)$ ; that is, each index is associated with one subgrouping with  $N_1$  and  $N_2$  samples, respectively. In fact, the CIP measures the information potential created by the dataset  $p(x)$  in places where samples of  $q(x)$  occur (or vice versa). One conventional way of measuring the distance between two clusters represented here by the random variables  $X_1$  and  $X_2$  is the average distance between pairs of samples, one in each cluster given by

$$\bar{D}_2(X_1, X_2) = \frac{1}{N_1 N_2} \sum_{x_i \in X_1} \sum_{x_j \in X_2} \|x_i - x_j\|^2. \quad (7.6)$$

This measure works well when the clusters are well separated and compact, but it will fail if the clusters are close to each other producing nonlinear boundaries. A better way of measuring the distance between clusters is to weight nonlinearly the distance between samples. The issue is to find a reasonable nonlinear weighting function. The information potential concept suggests that the weighting should be a localized symmetric window given by the kernel (e.g., Gaussian kernel). When the kernel function is selected, the average distance function becomes

$$\bar{D}_{CIP}(X_1, X_2) = \frac{1}{N_1 N_2} \sum_{x_i \in X_1} \sum_{x_j \in X_2} G_\sigma(\|x_i - x_j\|^2) \quad (7.7)$$

which is exactly the CIP.

### Relation of Cross Information Potential with Divergences

The concept of dissimilarity is central in clustering, and here CIP is motivated from the point of view of divergence measures. The Cauchy–Schwarz divergence measure (Chapter 2) is

$$D_{CS}(p, q) = -\ln \left( \frac{\int p(x) \bullet q(x) dx}{\sqrt{\int p^2(x) dx \int q^2(x) dx}} \right) = -\ln \left( \frac{V(p, q)}{\sqrt{\int p^2(x) dx \int q^2(x) dx}} \right). \quad (7.8)$$

Because the denominator is a product of two numbers that convey global information about the cluster shape, it serves as a normalization, therefore Eq (7.8) shows that in fact the numerator controls the behavior of  $D_{CS}(p, q)$ . In optimization, the fact that the minimum distance is not zero is irrelevant, so the *clustering evaluation function* was defined as the numerator of the CIP [114]

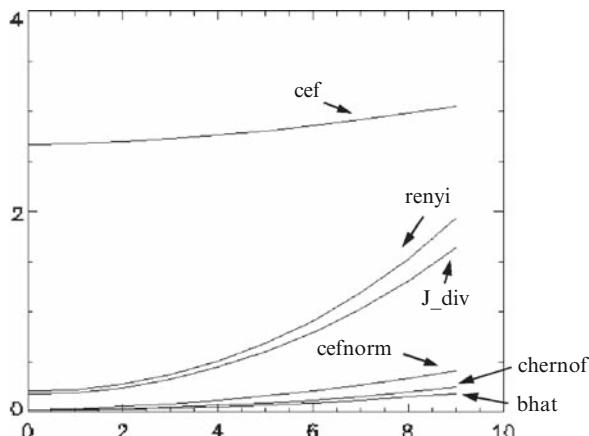
$$D_{CEF}(p, q) = -\ln V(p, q). \quad (7.9)$$

Recall that  $D_{CEF}$  has an information-theoretic interpretation as Renyi's quadratic cross-entropy. Moreover,  $D_{CEF}$  is a "pseudodistance" (because the identity property is not met), but it can be utilized as a figure of merit for clustering. Using the definition of the CIP it is easy to show that  $D_{CEF}$  is indeed always greater than zero and it is symmetric.

Writing the CIP as the integral of the product of  $p(x)$  and  $q(x)$  also shows that the difference between Eq (7.9) and the Bhattacharyya distance of Eq. (2.84) is just the square root in the argument. Again note that when  $D_{CEF}$  is used as a cost function in optimization the logarithm is not required because it is a monotonic function so we can say that the stationary points of  $V(p, q)$  and  $D_{CEF}(p, q)$  coincide. When optimizing the ratio in Eq. (7.8), we can think that the denominator works as a constraint for the optimization of the numerator, and consequently the optimization of  $V(p, q)$  coincides only with a special case of  $D_{CS}(p, q)$ .

Figure 7.2 compares  $D_{CEF}(p, q)$  experimentally with other conventional dissimilarity measures assuming that clusters are unimodal. The comparison is done between two Gaussian distributions, when the mean of one of the Gaussians is changed. The  $D_{CEF}$ ,  $D_{CS}$ , and Bhattacharyya are calculated with a kernel size of  $\sigma = 0.3$ , whereas the Chernoff distance of Eq. (2.85) is calculated with  $\sigma = 0.2$ , and Renyi's divergence in Eq. (2.73) for  $\alpha = 2$  and the  $J$  divergence of Eq. (1.11) is calculated with  $\sigma = 1.1$ .

The figure shows that although the minimum of  $D_{CEF}(p, q)$  is not zero when the two PDF functions are equal, its behavior as a function of cluster separation is consistent with the other measures, hence it can be used as a cost function in optimization. Many more tests are conducted in [114] comparing these distance measures, and we conclude that the product measures (Bhattacharyya and  $D_{CEF}$ ) are superior to the ratio measures (KL and Renyi)



**Fig. 7.2.** Comparison of different measures as a function of cluster separation (from [114]).

in experiments with few data samples. We also stress the fact that  $D_{CEF}(p, q)$  is by far the measure that displays the smallest algorithmic complexity when compared with the others due to the efficient computation of the information potential using, for instance, the incomplete Cholesky decomposition (batch mode) or the recursive IP (see Chapters 2 and 4) which are  $O(N)$ . This makes it practical for machine learning algorithms.

## Membership Function for Multiple Clusters

The previous definition of CIP was given for two clusters, and this section generalizes the definition for more clusters. To be more explicit and simplify the evaluations, a membership function is included in Eq. (7.5) which explicitly shows sample partnership. For the two cluster case, every sample is tagged with a membership function  $\mathbf{m}$  defined as:

$$\text{If } x_i \text{ belongs to cluster } C_1, \quad \mathbf{m}_i = [1, 0]^T \quad (7.10)$$

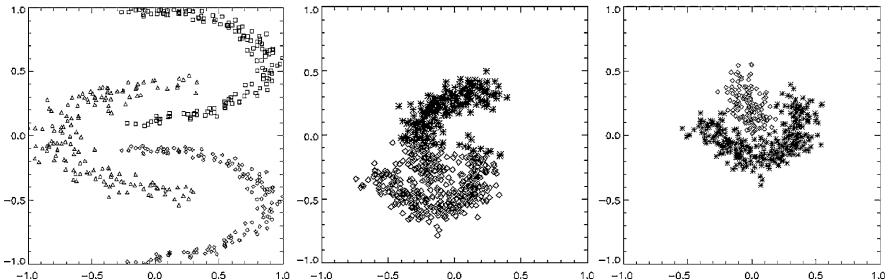
( $\mathbf{m}_i = [0, 1]^T$  when  $x_i$  belongs to  $C_2$ ).  $\mathbf{m}_i(k)$   $k = 1, 2$  denotes the element  $k$  of  $\mathbf{m}_i$ , and notice that now the IP becomes a function of memberships. Redefine the CIP as

$$\hat{V}(\mathbf{m}_1, \dots, \mathbf{m}_N) = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N (1 - \mathbf{m}_i^T \mathbf{m}_j) G_{\sigma\sqrt{2}}(x_i - x_j). \quad (7.11)$$

In the case of  $K$  clusters, when  $x_i$  belongs to cluster  $C_1$ ,  $\mathbf{m}_i = [1, 0, \dots, 0]^T$  (the size of the binary vector is  $K$ , so this notation is easily generalized for multiple clusters. The stepwise nature of the function  $\hat{V}(\mathbf{m}_1, \dots, \mathbf{m}_N)$  creates problems for gradient-based methods that we have been able to resolve (see Section 7.5).

## Case Study: CEF for Reclustering

The CEF can be used for reclustering the results of a traditional clustering algorithm as k-means with the goal of further decreasing the CEF. The original algorithm used greedy search (see [114]) which is slow and does not scale up, but it is always able to find the best possible solution. Figure 7.3 shows the results of clustering the three datasets in [114] (triangles, crosses and circles for each cluster) using the CEF algorithm initialized with three and two clustering centers, respectively. These datasets were designed to require nonlinear discriminant functions (MLPs or RBFs) in a supervised paradigm, but all display a valley between the clusters. We used a kernel size ranging from 0.005 to 0.02, when the data are normalized between  $-1$  and  $+1$ . The kernel size  $\sigma$  is important to produce the best assignments, but we found that the clustering is mildly sensitive to the actual value, provided it is in the correct range, that is, in the range where the PDF estimation is sensitive to



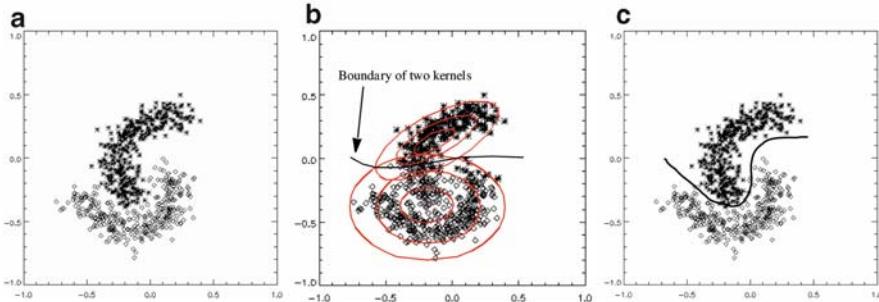
**Fig. 7.3.** Clustering with CEF (left to right:  $\sigma^2 = 0.05$ ,  $\sigma^2 = 0.021$ ,  $\sigma^2 = 0.026$ ) (from [114]).

the structure of the data. We should realize that neither k-means nor Gaussian mixtures would divide the datasets in the way the CEF did. In the figure, the number of samples per cluster is 150 points for the left panel, 350 samples for the middle panel, and the right panel has dissimilar number of 350 and 150 points. In the three class problem there is little overlap among the clusters, although the problem is hard due to the nonconvex boundaries. The two cases of two-class problems have substantial overlap and even an uneven number of points in the last example.

CEF always produces good clustering with smooth convergence although the best results are obtained with slightly different kernel sizes. Against our expectation, CEF is not too sensitive to the unequal size of the clusters in the right most example [114]. The convergence of the algorithm is normally smooth. In the early portion of the training, the number of samples that changed cluster was  $N/8 - N/4$  and it decreased to about 1 to 2 per iteration towards the end. Samples that stand alone in the data space (outliers) pose a problem to the algorithm, because they tend to define their own clusters, and were aggregated to the closest cluster in post processing.

### Comparison with other Supervised and Unsupervised Techniques

We compared the CEF procedure with the well-known and widely used expectation maximization (EM) algorithm, and with a single hidden layer MLP (ten hidden units). The MLP was trained with backpropagation (early stopping) in a different 1000 sample set obtained from the same data model. The k-means (not shown) could not separate the data as can be expected, because its cost function uses minimum variance. The CEF procedure, the EM and the MLP had perfect performance in the first dataset (the three moons) of Figure 7.3. The results in the second data set of Figure 7.3 are depicted in Figure 7.4 (CEF, EM with two clusters and the MLP, respectively) and visually the CEF achieves a clustering very similar to the MLP. The EM algorithm can actually capture the data structure better if more than two clusters are selected, but how to aggregate these clusters remains problematic.



**Fig. 7.4.** (a) Clustering with CEF, (b) mixture of Gaussians, and (c) MLP, respectively (from [114]).

**Table 7.1.** Confusion Tables: Left MLP, Right CEF (From [114])

DATA SET 2	Class 1	Class 2	DATA SET 2	Class 1	Class 2
Class 1	342	8	Class 1	340	10
Class 2	9	341	Class 2	12	338
DATA SET 3	Class 1	Class 2	DATA SET 3	Class 1	Class 2
Class 1	145	5	Class 1	144	6
Class 2	9	341	Class 2	9	341

**Table 7.2.** Confusion Matrix of Iris Data

	Class 1	Class 2	Class 3
Class 1	50	0	0
Class 2	0	42	8
Class 3	0	6	44

Table 7.1 compares the MLP and CEF performance in the two datasets of Figure 7.3. From the table we can conclude that the CEF initialized with two clusters performs at the same level as the MLP, although the CEF is an unsupervised learning method and the MLP uses class label information. Notice also that the results are achieved with a careful selection of the kernel size and an exhaustive search, which in practice does not scale to a high number of samples and requires the development of a learning algorithm. Still it shows the power of the CEF cost to exploit the structure of the clusters.

The second test uses the well-known Iris data, which consist of three different classes with a four-dimensional feature vector, and 50 samples for each class. We should mention that a trained perceptron makes 19 mistakes, where a trained single hidden layer MLP makes 3 mistakes [253]. From Table 7.2 we conclude that given the exact number of clusters and the right kernel size, the CEF clustering is able to refine the clustering produced by K means

to the point that it competes with supervised techniques. CEF seems to be insensitive to the shape of the clusters, but to work properly it still requires a “valley” among the clusters (e.g., a region of smaller sample density between the clusters).

## 7.5 A Gradient Algorithm for Clustering with D<sub>CS</sub>

Now that the power of the clustering evaluation function was briefly analyzed, it is important to provide a general-purpose algorithm for clustering that exploits its good properties and can be applied directly to the data, not to a pre-clustered configuration. This effort falls in the family of kernel-based clustering algorithms but avoids direct PDF estimation due to the use of the information potential estimator. In recent years, many kernel-based global optimization algorithms have been proposed, for example, support vector machine inspired algorithms [30], graph-theoretic algorithms [296], and spectral clustering methods [230]. Because the CEF is the logarithm of the cross-information potential of ITL, we can either use the Euclidean or the Cauchy-Schwarz divergences estimated by the information potential directly from the data. Here we optimize the Cauchy-Schwarz divergence (D<sub>CS</sub>) of Eq. (7.8) by means of the Lagrange multiplier formalism using kernel size annealing, which exhibits a well-known robustness property with regard to avoiding local optima of the cost function.

We derive a fuzzy clustering approach to optimize D<sub>CS</sub> under constraints with regard to the values the membership functions can take. This approach is basically a constrained gradient descent procedure, and can be interpreted as a gradient search with built-in variable step sizes for each coordinate direction in feature space. We solve the optimization problem with respect to the membership functions, using the Lagrange multiplier formalism. The problem with most kernel-based methods is that it is not clear how to choose a specific kernel size for optimal performance. We circumvent this problem to some degree by incorporating kernel size annealing during the adaptation as mentioned already in Chapter 3 (Section 3.4) that has the potential to provide a global optimum provided the annealing rate is “sufficiently slow”. However, the smallest kernel size still needs to be set from the data.

This algorithm is gradient-based, thus the log can be eliminated in  $D_{CS}(p, q) = -\log J_{CS}(p, q)$  and work directly with the argument

$$J_{CS}(p, q) = \frac{\int p(x)q(x)dx}{\sqrt{\int p^2(x)dx \int q^2(x)dx}} \quad (7.12)$$

and the corresponding estimator using the information potential becomes

$$\hat{J}_{CS}(p, q) = \frac{\frac{1}{N_p N_q} \sum_{i=1}^{N_p} \sum_{j=1}^{N_q} G_{\sigma\sqrt{2}}(x_i - x_j)}{\sqrt{\frac{1}{N_p^2} \sum_{i=1}^{N_p} \sum_{i'=1}^{N_p} G_{\sigma\sqrt{2}}(x_i - x_{i'}) \frac{1}{N_q^2} \sum_{j=1}^{N_q} \sum_{j'=1}^{N_q} G_{\sigma\sqrt{2}}(x_j - x_{j'})}}. \quad (7.13)$$

For each scalar (for simplicity) data pattern  $x_i$ ,  $i = 1, \dots, N$ ,  $N = N_p + N_q$ , we use the corresponding membership vector  $\mathbf{m}_i$  in the derivation and Eq. (7.13) can be rewritten as

$$\hat{J}_{CS}(\mathbf{m}_1, \dots, \mathbf{m}_N) = \frac{\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N (1 - \mathbf{m}_i^T \mathbf{m}_j) G_{\sigma\sqrt{2}}(x_i - x_j)}{\sqrt{\sum_{i=1}^N \sum_{j=1}^N m_{i1} m_{j1} G_{\sigma\sqrt{2}}(x_i - x_j) \sum_{i=1}^N \sum_{j=1}^N m_{i2} m_{j2} G_{\sigma\sqrt{2}}(x_i - x_j)}}. \quad (7.14)$$

Here we are assuming a Gaussian kernel and that the data are one-dimensional, but the form of the expressions is unchanged except that now the kernel is multidimensional and normally circularly symmetric of s.d.  $\sqrt{2}\sigma$ . In the case of multiple clusters,  $C_k$ ,  $k = 1, \dots, K$ , Eq. (7.14) becomes

$$\hat{J}_{CS}(\mathbf{m}_1, \dots, \mathbf{m}_N) = \frac{\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N (1 - \mathbf{m}_i^T \mathbf{m}_j) G_{\sigma\sqrt{2}}(x_i - x_j)}{\sqrt{\prod_{k=1}^K \sum_{i=1}^N \sum_{j=1}^N m_{ik} m_{jk} G_{\sigma\sqrt{2}}(x_i - x_j)}}. \quad (7.15)$$

where each  $m_i$  is a binary  $K$ -dimensional vector. Only the  $k$ th element of any  $m_i$  equals one, meaning that the corresponding data pattern  $x_i$  is assigned to cluster  $k$ .

At this point we note some particularly interesting features of the Cauchy–Schwarz divergence. Friedman and Tukey [105] defined a measure of cluster compactness identical to the information potential. This is in contrast to the Euclidean compactness measure defined by the sum-of-squares error. Similarly, the CIP Eq. (7.5) can be considered a measure of the inter cluster compactness. According to Eq. (7.14), it is clear that in order for  $J_{CS}$  to be small, the inter cluster compactness should be small, and the product of intra cluster compactness should be large. This means that in order for  $D_{CS}$  to be large, the sum of the entropies of the individual clusters must be small, while at the same time the cross-entropy between the clusters must be large. This makes perfect sense. Finally, it should be mentioned that we assume a priori knowledge about the number  $K$  of clusters inherent in the dataset (see [163] for methods to choose the number of clusters). We conclude that Eq. (7.14) and (7.15) provide an information-theoretic cost function for clustering, capable of capturing data structure beyond mere second-order statistics to which many traditional clustering cost functions are restricted.

## Optimization of DCS by Lagrange Multipliers

For a given dataset consisting of the data patterns,  $x_i$ ,  $i = 1, \dots, N$ , each data pattern is assigned to a crisp membership with respect to the  $K$  clusters, represented by the membership vector  $\mathbf{m}_i$ . Our goal is to assign memberships to samples such that  $\hat{J}_{CS}(\mathbf{m}_1, \dots, \mathbf{m}_N)$  is minimized, because this corresponds to the  $D_{CS}$  being maximized. According to [163] this optimization problem benefits from the method of Lagrange multipliers. Because this is a technique of differential calculus,  $\hat{J}_{CS}(\mathbf{m}_1, \dots, \mathbf{m}_N)$  is made continuous and differentiable by fuzzifying the membership vectors. This approach is sometimes called fuzzy clustering [315]. Our first step is to let the elements of  $\mathbf{m}_i$  to be in  $[0,1]$ ,  $i = 1, \dots, N$ , and to define the following constrained optimization problem:

$$\min_{m_1, \dots, m_N} \hat{J}_{CS}(\mathbf{m}_1, \dots, \mathbf{m}_N) \quad \text{subject to } \mathbf{m}_j^T \mathbf{1} - 1 = 0, \quad j = 1, \dots, N \quad (7.16)$$

where  $\mathbf{1}$  is a  $K$ -dimensional vector whose elements are all one. Hence, a data pattern is allowed to have a certain degree of membership in any cluster, but the constraint ensures that the sum of the memberships adds up to one. Now we make a convenient change of variables. Let  $\mathbf{m}_i(k) = v_i^2(k)$ ,  $k = 1, \dots, K$ , and consider the following optimization

$$\min_{v_1, \dots, v_N} \hat{J}_{CS}(\mathbf{v}_1, \dots, \mathbf{v}_N) \quad \text{subject to } \mathbf{v}_j^T \mathbf{v} - 1 = 0, \quad j = 1, \dots, N \quad (7.17)$$

The constraints for the problem stated in Eq. (7.17) with regard to  $\mathbf{v}_j$ ,  $j = 1, \dots, N$ , are equivalent to the constraints for the problem stated in Eq. (7.16) with regard to  $\mathbf{m}_j$ . The optimization problem, Eq. (7.17), amounts to adjusting the vectors  $\mathbf{v}_i$ ,  $i = 1, \dots, N$ , such that

$$\frac{\partial \hat{J}_{CS}}{\partial \mathbf{v}_i} = \left( \frac{\partial \hat{J}_{CS}^T}{\partial \mathbf{m}_i} \frac{\partial \mathbf{m}_i}{\partial \mathbf{v}_i} \right)^T = \Gamma \frac{\partial \hat{J}_{CS}}{\partial \mathbf{m}_i} = 0, \quad (7.18)$$

where  $\Gamma = \text{diag}(2\sqrt{m_{i1}}, \dots, 2\sqrt{m_{iK}})$  which is very interesting. Notice that if all of the diagonal elements  $\sqrt{m_{iK}}$  are positive, the direction of the gradients of  $\partial \hat{J}_{CS} / \partial v_i$  and  $\partial \hat{J}_{CS} / \partial \mathbf{m}_i$  will always be the same. Hence, in this case, these scalars can be thought of as variable step sizes built into the gradient descent search process, as a consequence of the change of variables. We force all the elements of the membership vectors  $m_i$  to always be positive, by adding a small positive constant  $\alpha$  (e.g.  $\alpha \sim 0.01$ ) to all the elements during each membership update in the iterative algorithm. This approach also has the effect of introducing a small amount of noise into the algorithm, a strategy that is well known as an additional means to help avoid local optima.

The computation of all the gradients  $\partial \hat{J}_{CS} / \partial \mathbf{m}_i$ ,  $i = 1, \dots, N$ , is an  $O(N^2)$  procedure at each iteration. Let us write Eq. (7.15) as  $J_{CS} = U/V$  where

$$U = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \left(1 - \mathbf{m}_i^T \mathbf{m}_j\right) G_{\sqrt{2}\sigma}(x_i - x_j) \quad \text{and} \quad V = \sqrt{\prod_{k=1}^K v_k(\mathbf{m}_1, \dots, \mathbf{m}_N)} \quad (7.19)$$

with  $v_k = \sum_{i=1}^N \sum_{j=1}^N \mathbf{m}_i(k) \mathbf{m}_j(k) G_{\sqrt{2}\sigma}(x_i - x_j)$ . Hence

$$\begin{aligned} \frac{\partial \hat{J}_{CS}}{\partial \mathbf{m}_i} &= \frac{V \frac{\partial U}{\partial \mathbf{m}_i} - U \frac{\partial V}{\partial \mathbf{m}_i}}{V^2} \quad \text{with} \quad \frac{\partial U}{\partial \mathbf{m}_i} = - \sum_{j=1}^N \mathbf{m}_j G_{\sqrt{2}\sigma}(x_i - x_j), \\ \frac{\partial V}{\partial \mathbf{m}_i} &= \frac{1}{2} \sum_{k'=1}^K \sqrt{\frac{\prod_{\substack{k=1 \\ k \neq k'}}^K v_k}{v_{k'}}} \frac{\partial v_{k'}}{\partial \mathbf{m}_i}, \end{aligned} \quad (7.20)$$

where  $\frac{\partial v_{k'}}{\partial m_i} = \left[0 \dots 2 \sum_{j=1}^N m_j(k') G_{\sqrt{2}\sigma}(x_i - x_j), \dots, 0\right]^T$ . Thus, only the element  $k'$  of this vector is nonzero. The necessary conditions that the solution of Eq. (7.17) must obey are commonly generated by constructing a function,  $L = L(\mathbf{v}_1, \dots, \mathbf{v}_N, \lambda_1, \dots, \lambda_N)$  known as the Lagrange function, given by;

$$L = \hat{J}_{CS}(\mathbf{v}_1, \dots, \mathbf{v}_N) + \sum_{j=1}^N \lambda_j (\mathbf{v}_j^T \mathbf{v}_j - 1) \quad (7.21)$$

where  $\lambda_j$ ,  $j = 1, \dots, N$ , are the Lagrange multipliers. The necessary conditions for the extremum of  $L$ , which also corresponds to the solution of the original problem in Eq. (7.16), are given by (note that  $\mathbf{v}_j^T \mathbf{v}_j - 1 = 0$ )

$$\begin{cases} \frac{\partial L}{\partial \mathbf{v}_i} = \frac{\partial \hat{J}_{CS}}{\partial \mathbf{v}_i} + \sum_{k=1}^N \lambda_k \frac{\partial (\mathbf{v}_k^T \mathbf{v}_k - 1)}{\partial \mathbf{v}_i} = 0 \\ \frac{\partial L}{\partial \lambda_j} = \mathbf{v}_j^T \mathbf{v}_j - 1 \quad j = 1, \dots, N. \end{cases} \quad (7.22)$$

From Eq. (7.22) the vector  $\mathbf{v}_i$  is derived as follows,

$$\frac{\partial \hat{J}_{CS}}{\partial \mathbf{v}_i} + 2\lambda_i \mathbf{v}_i = 0 \Rightarrow \mathbf{v}_i^+ = -\frac{1}{2\lambda_i} \frac{\partial \hat{J}_{CS}}{\partial \mathbf{v}_i} \quad i = 1, \dots, N \quad (7.23)$$

where the superscript  $+$  denotes the updated vector. We solve for the Lagrange multipliers,  $\lambda_i$  by evaluating the constraints given by Eq. (7.23) as follows,

$$\mathbf{v}_i^{+T} \mathbf{v}_i^+ - 1 = 0 \Rightarrow \lambda_i = \frac{1}{2} \sqrt{\frac{\partial \hat{J}_{CS}^T}{\partial \mathbf{v}_i} \frac{\partial \hat{J}_{CS}}{\partial \mathbf{v}_i}}. \quad (7.24)$$

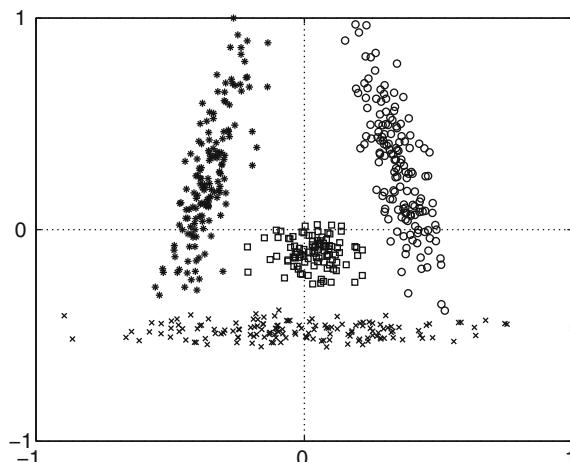
After convergence of the algorithm, or after a predetermined number of iterations, the maximal element of each membership vector  $m_i$  is set to one, and the rest to zero.

We propose to initialize the vectors as  $\mathbf{v}_i = |N(0; \gamma^2 \mathbf{I})|$ , where  $N$  denotes the Gaussian distribution and  $\gamma$  is small (e.g., 0.5). It is clear that the vectors  $v_i$ , do not initially obey the constraint of Eq. (7.17). We have observed that after the first iteration, the constraint is always obeyed. This algorithm is order-independent: the order in which the data patterns are presented to the algorithm is of no importance. The computational complexity is  $O(N^2)$  for each iteration through the whole dataset. The increase in complexity compared to k-means, which is  $O(N)$ , is the price we have to pay in order to capture higher order statistical properties of the data.

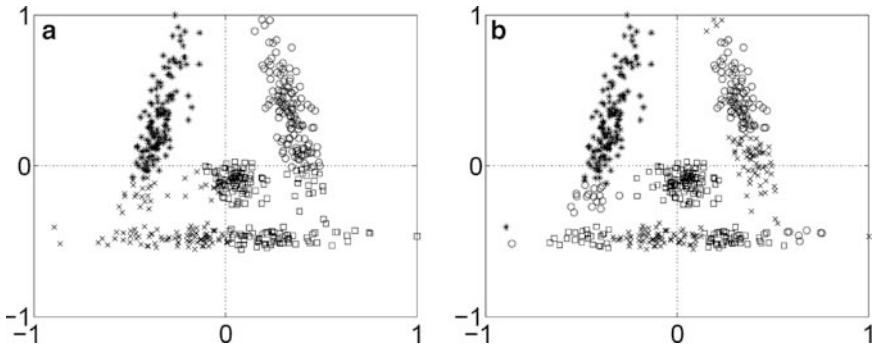
### Learning by Kernel Annealing

In this chapter we use extensively a synthetic data set built from four Gaussian anisotropic clusters shown in Figure 7.5 with the labels that created the clusters. The size of the kernel is the only free parameter of this ITL fuzzy algorithm (except for the constant  $\gamma$ , discussed in the previous section) as is normally the case in kernel methods, but the issue is how to choose a suitable kernel size for good results.

The learning strategy proposed here consists in allowing the kernel size to decrease over time as the iterative Lagrange multiplier optimization algorithm proceeds as explained in Chapters 3, 5 and 6. This means that an upper limit for the size of the kernel, a lower limit, and an annealing rate need to be determined for the problem, increasing the number of free parameters to be set by the experimenter. However, we will show that the resulting algorithm is not too sensitive to the actual value of each of these parameters, and that



**Fig. 7.5.** Artificially created dataset used in kernel annealing case study. All clusters consist of 150 patterns, except the cluster in the middle, consisting of 100 patterns (squares).



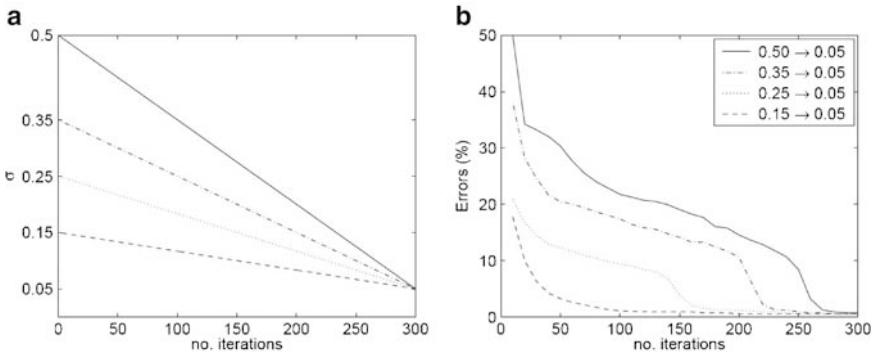
**Fig. 7.6.** Result of clustering experiment using static kernels of size (a)  $\sigma = 0.35$  and (b) 0.05, respectively. The result is not satisfying in either case (from [162]).

this learning strategy greatly helps reduce the risk of being trapped in a local minimum of the performance surface. The end result is that kernel annealing makes the algorithm much more flexible and robust. However, for datasets where the density of the clusters is grossly different, it is difficult to tune the annealing rate.

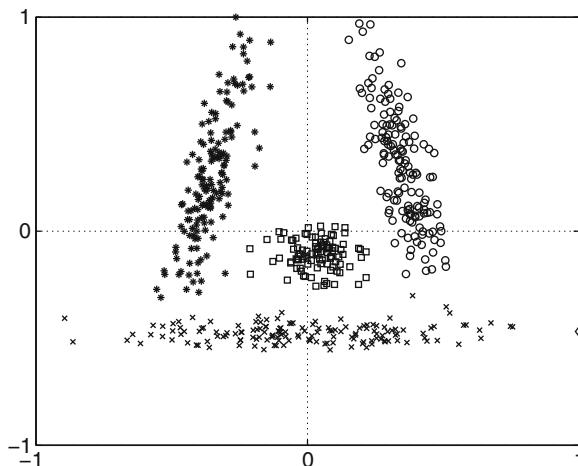
We illustrate the annealing property of our algorithm in the dataset of Figure 7.5. First we apply the ITL fuzzy clustering algorithm to this dataset using a fixed kernel size and  $K = 4$  clusters (Figure 7.6a).

The membership vectors are initialized as proposed, and the constant  $\alpha$  is set to 0.01. In the first experiment the algorithm is run a total of 20 times, using a kernel of size  $\sigma = 0.35$ . The number of iterations for each trial is fixed at 300, and the resulting labels are evaluated. In every one of these trials the algorithm quickly converges to a k-means-like solution, clearly wrong, from which it only sporadically recovers. Similarly, in the second experiment (Figure 7.6b), the kernel-size is fixed to  $\sigma = 0.05$ . For such a relatively small kernel size, the algorithm is not able to unravel the global structure of the data, but produces clusters where the parts of the dataset which are designated with the same label, may be located far from each other.

Now we demonstrate the result of annealing the kernel over 300 iterations. Four experiments were performed, each with a different upper limit for the kernel size, but all with the same lower limit, which is given by  $\sigma = 0.05$ . In Figure 7.7a, the size of the kernel versus the number of iterations is plotted for each annealing scheme. For each of the four experiments, 20 runs are attempted, and the mean percentage of errors plotted, evaluated at every tenth iteration. With annealing the results are quite remarkable; that is, the algorithm obtains a solution very close to the optimal solution in every single trial, as shown in Figure 7.7b. No matter how large the initial kernel is, provided it is annealed sufficiently slowly over iterations, the algorithm always escapes the k-means local minimum. When the initial kernel size is started at  $\sigma = 0.35$ , 0.25, and 0.15, respectively, the end result is the same (three errors only in



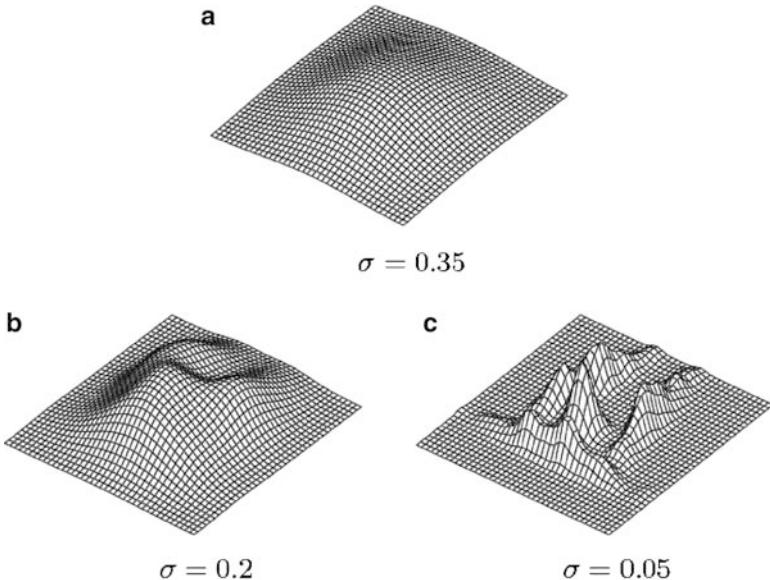
**Fig. 7.7.** The clustering result is near optimal in all 300 trials. The annealing rates are shown in (a), and (b) displays the mean percentage of clustering errors for each rate.



**Fig. 7.8.** Typical example of clustering result using kernel annealing. There are three errors in the lower-right corner corresponding to 0.55% errors.

the lower right corner of Figure 7.8), in every single trial. This corresponds to an error rate of 0.55%. For  $\sigma = 0.5$  as the starting point, the end result is four errors in 19 of the 20 trials, and three errors in the remaining trial. We can see that for large kernel sizes, the error percentage decreases slowly until the kernel size reaches about  $\sigma = 0.2$ .

Finally, Figure 7.9a, b, and c show the underlying Parzen PDF estimates corresponding to a kernel size of  $\sigma = 0.35$ , 0.2, and 0.05, respectively. The smoothing effect inherent in the Parzen kernel method is clearly visible for the PDF estimate corresponding to  $\sigma = 0.35$ , but for  $\sigma = 0.2$ , still no clear structure in the dataset can be observed. Even so, this kernel size seems to be a critical value for the annealing process for this dataset. The PDF estimate for  $\sigma = 0.05$  clearly shows the structure of the data.



**Fig. 7.9.** Different looks at the dataset with different kernelsizes.

How to select a particular annealing scheme is still an open question. However, our experience recommends selecting the upper limit for the kernelsize such that the width of the kernel covers at least about 25% of the samples present in the dataset. For the lower limit, it seems appropriate to select the size of the kernel such that it covers about 5% of the data. In most cases it seems sufficient to anneal over about 300 to 400 iterations to guarantee robust performance over all clustering trials.

### Case Studies with D<sub>CS</sub> Clustering

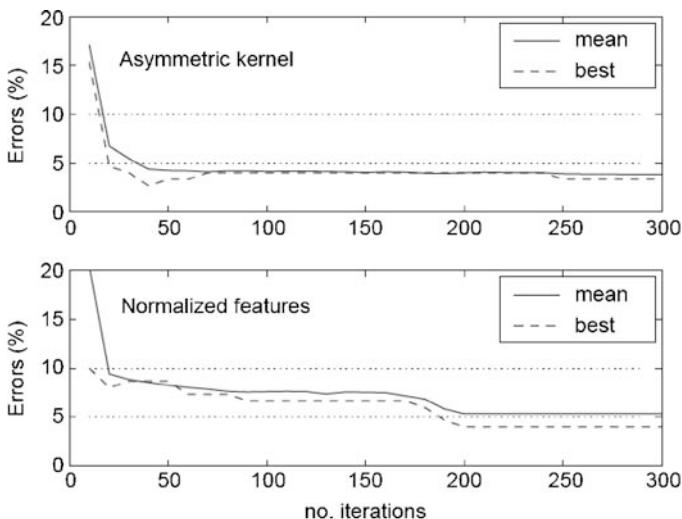
We now demonstrate that the fuzzy clustering algorithm is able to unravel the underlying structure of several different datasets, and hence to produce reasonable clusters. The datasets are real datasets extracted from the UCI repository, University of California, Irvine [227]. Special attention is paid to the IRIS dataset, inasmuch as this is a well-known benchmark dataset for clustering. In all cases the membership vectors were initialized as was proposed in Section 7.5. The constant  $\gamma$  is set to 0.01, and the comparison with the traditional k-means algorithm is conducted.

The clustering of the IRIS dataset achieves 9.3% misclassifications, averaged over 20 trials, and for a wide range of kernel sizes and annealing rates. This relatively unsatisfying result can be attributed to the symmetric kernel used in the D<sub>CS</sub> algorithm, which does not perform well when the input dynamic ranges are different. In such cases two simple methods can

be used to preprocess the data. The first method applies the CS clustering method with an asymmetric kernel. For a multidimensional asymmetric Gaussian kernel with a diagonal covariance matrix  $\Sigma$ , each diagonal element  $\Sigma_i$  is calculated for feature number  $i$  by the Sheather–Jones plug-in method [295]. Optimality is defined by the mean integrated squared error between the true PDF and the one-dimensional Parzen estimate. The calculation of the asymmetric kernel using the IRIS dataset yields the following bandwidths.  $\Sigma_1 = 0.32$ ,  $\Sigma_2 = 0.14$ ,  $\Sigma_3 = 0.19$  and  $\Sigma_4 = 0.08$ .

The second method simply normalizes the dynamic ranges of the features one-by-one, such that they lie within the same interval, in our case  $[-1, 1]$ . Thereafter clustering can use a symmetric kernel. The kernel size annealing is performed such that the mean of these values decreases from approximately 0.5 to approximately 0.05. In the experiment using normalized features, we anneal the kernel size from  $\Sigma = 0.5$  to  $\Sigma = 0.05$ . In both cases the annealing is performed over 300 iterations.

Figure 7.10 shows the mean error percentage over 20 trials (solid curve) along with the best result (dashed curve), using the asymmetric kernel and the normalized features, respectively. The results are more satisfying, taking into consideration that the IRIS dataset is known to be difficult to cluster because of a nonlinear boundary between two of the three clusters. In the case of using the asymmetric kernel, the best result yields only 3.33% errors. When normalizing the features, the best result yields 4% errors. Our results of clustering the



**Fig. 7.10.** Clustering the IRIS dataset. The upper panel shows the mean result and the best result with CS-clustering and an asymmetric kernel. Similarly, in the lower panel, the result using normalized features is shown. The kernel is annealed from  $\sigma = 0.5$  to  $\sigma = 0.05$  over 300 iterations.

IRIS dataset compares favorably to most results achieved by recent clustering methods on the same dataset (see e.g., [269], [114], [30], [296]).

## 7.6 Mean Shift Algorithms and Renyi's Entropy

Let us consider a dataset  $X = \{x_1, \dots, x_N\}, x \in R^D$  with independent and identically distributed (i.i.d.) samples. Using the nonparametric Parzen window method, the probability density estimate is given by  $\hat{p}_{X,\sigma}(x) = 1/N \sum_{i=1}^N G_\sigma(x - x_i)$  where  $G_\sigma$  is a Gaussian kernel with bandwidth  $\sigma > 0$ . The modes of the PDF are solutions of the equation  $\nabla p_{X,\sigma}(x) = 0$  [109]. Substituting the estimator and rearranging the gradient equation into an iterative fixed-point equation, we obtain

$$x(n+1) = m(x(n)) = \frac{\sum_{i=1}^N G_\sigma(x - x_i)x_i}{\sum_{i=1}^N G_\sigma(x - x_i)}. \quad (7.25)$$

The term  $m(x) - x$  was coined “mean shift” by Fukunaga and Hostetler in their landmark paper [109]. Given an initial dataset  $X(0) = X_o$  and using Eq. (7.25), we successively “blur” the dataset  $X_o$  to produce datasets  $X(1), \dots, X(n)$ . As the new datasets are produced the algorithm forgets the previous one which gives rise to the blurring process. It was Cheng [55] who first pointed out this and renamed the fixed point update Eq. (7.25) as the *blurring mean shift*.

This successive blurring collapses the data rapidly and hence made the algorithm unstable. In his 1995 paper, which sparked renewed interest in mean shift, Cheng proposed a modification in which two different datasets would be maintained, namely  $X$  and  $X_o$  defined as  $X_o = \{x_{o1}, \dots, x_{oN}\}, x_o \in R^D$ . The dataset  $X$  would be initialized to  $X_o$ . At every iteration, a new dataset  $X(n+1)$  is produced by comparing the present dataset  $X(n)$  with  $X_o$ . Throughout this process  $X_o$  is fixed and kept constant. This stable fixed-point update is called the mean shift algorithm and is given by

$$x(n+1) = m(x(n)) = \frac{\sum_{i=1}^{N_o} G_\sigma(x - x_{oi})x_{oi}}{\sum_{i=1}^{N_o} G_\sigma(x - x_{oi})}. \quad (7.26)$$

To be consistent with the existing mean shift literature, we call these algorithms Gaussian blurring mean shift (GBMS) and Gaussian mean shift (GMS), respectively, indicating the use of Gaussian kernels specifically.

Recent advancements in Gaussian mean shift has made it increasing popular in the image processing and vision communities. First, this class of algorithms can be used for clustering and they do not require the specification of the number of clusters. Second, they are also very fast due to the fixed-point iteration, and they do not need a step size. In particular, the mean shift vector of GMS has been shown to always point in the direction of normalized

density gradient [55]. Points lying in low-density regions have a small value of  $p(x)$ , so the normalized gradient at these points has large value. This helps the samples to quickly move from low density regions towards the modes. On the other hand, due to the relatively high value of  $p(x)$  near the mode, the steps are highly refined around this region. This adaptive nature of step size gives GMS a significant advantage over traditional gradient based algorithms where the step size selection is a well-known problem.

A rigorous proof of stability and convergence of GMS was given by Comaniciu and Meer [61] where they proved that the sequence generated by Eq. (7.26) is a Cauchy sequence that converges due to the monotonic increasing sequence of the PDFs estimated at these points. Furthermore, the trajectory is always smooth in the sense that the consecutive angles between mean shift vectors is always between  $(-\pi/2, \pi/2)$ . Carreira-Perpiñán [50] also showed that GMS is an EM algorithm and thus has a linear convergence rate (unlike gradient descent which is sublinear). Due to these interesting and useful properties, GMS has been successfully applied in low level vision tasks such as image segmentation and discontinuity preserving smoothing [61] as well as in high level vision tasks such as appearance-based clustering [256] and real-time tracking of non rigid objects [60]. Carreira-Perpiñán [49] used mean shift for mode finding in a mixture of Gaussian distributions. The connection to the Nadarayana-Watson estimator from kernel regression and the robust M-estimators of location has been thoroughly explored by Comaniciu and Meer [61]. With just a single parameter to control the scale of analysis, this simple nonparametric iterative procedure has become particularly attractive and suitable for a wide range of applications.

### Connection to Information and Cross-Information Potentials

There is a profound and interesting connection between mean shift algorithms and Renyi's entropy. Consider an original dataset  $X_o = \{x_{o1}, \dots, x_{oN}\}$ ,  $x_o \in R^D$  with i.i.d samples. This dataset is kept fixed throughout the experiment. Let us define another dataset  $X = \{x_1, \dots, x_N\}$ ,  $x \in R^D$  with initialization  $X = X_o$  and  $\sigma_X = \sigma_{X_o}$ . With this setup, consider the following cost function.

$$J(X) = \min_X H_2(X) = \max_X \log V(X). \quad (7.27)$$

Notice that  $X$  is the variable which evolves over time and hence appears in the argument of the cost function. Because the log is a monotone function we can redefine  $J(X)$  as

$$J(X) = \max_X V(X) = \max_X \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N G_\sigma(x_i - x_j). \quad (7.28)$$

This means that the maximization of the information potential provides a value equal to the maximum interaction among the samples dictated by the

kernel, that is, when all of them collapse to the same point in space (i.e., a delta function). In order to find where the delta function is located, we have to analyze how the samples move. Differentiating  $J(X)$  with respect to  $x_{k=\{1,2,\dots,N\}} \in X$  and equating it to zero gives

$$F(x_k) = \frac{1}{2\sigma^2 N} \sum_{j=1}^N G_\sigma(x_k - x_j)(x_j - x_k) = 0. \quad (7.29)$$

$F(x_k)$  is the information force acting on particle  $x_k$  due to all other samples within the dataset  $X$ . Thus we would like to evolve this dataset such that the samples reach an equilibrium position with net force acting on each sample equal to zero. Rearranging the above equation gives us the fixed-point update rule

$$x_k(n+1) = m(x_k(n)) = \frac{\sum_{j=1}^N G_\sigma(x_k - x_j)x_j}{\sum_{j=1}^N G_\sigma(x_k - x_j)}. \quad (7.30)$$

Comparing Eq. (7.30) to Eq. (7.25) we see that this is exactly equal to the GBMS algorithm. Thus GBMS minimizes the overall Renyi's quadratic entropy of the dataset. The only stationary solution of Eq. (7.30) is a single point, which is obtained by successive "blurring" of the initial dataset.

We can rectify this deficiency by making a slight modification to the cost function. Instead of minimizing Renyi's quadratic entropy we minimize the cross-entropy or maximize the CIP,  $V(X, X_o)$ , after dropping the logarithm

$$J(X) = \max_X V(X, X_o) = \max_X \frac{1}{NN_o} \sum_{i=1}^N \sum_{j=1}^{N_o} G(x_i - x_{oj}). \quad (7.31)$$

Now this method indeed places the delta function over the mode of the original data, as we can realize by differentiating  $J(X)$  with respect to  $x_{k=\{1,2,\dots,N\}} \in X$  and equating it to zero yields

$$F(x_k) = \frac{1}{2\sigma^2 N} \sum_{j=1}^N G_\sigma(x_k - x_{oj})(x_{oj} - x_k) = 0. \quad (7.32)$$

Thus in this scenario, the samples of dataset  $X$  move under the influence of the "cross" information force exerted by samples from dataset  $X_o$ . The fixed-point update is

$$x_k(n+1) = m(x_k(n)) = \frac{\sum_{j=1}^N G_\sigma(x_k - x_{oj})x_{oj}}{\sum_{j=1}^N G_\sigma(x_k - x_{oj})}. \quad (7.33)$$

Indeed, this is the GMS update equation as shown in Eq. (7.27). By minimizing the cross information potential, GMS evolves the dataset  $X$  and at the

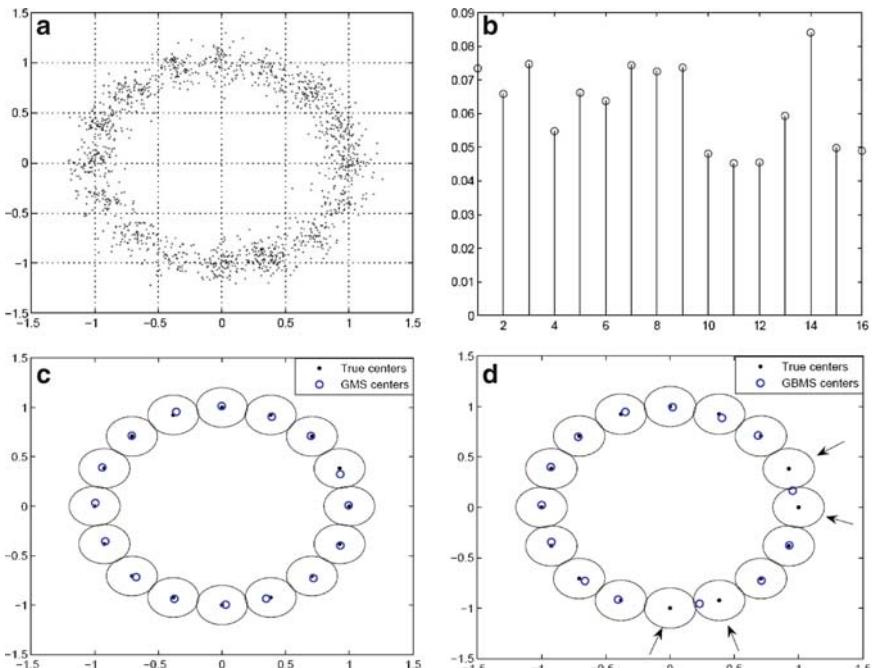
same time keeps in “memory” the original dataset  $X_o$ . Because  $F(x, X_o) \propto \nabla p_{X_0, \sigma}(x)$ , the result is movement of the samples  $x_{k=\{1,2,\dots,N\}} \in X$  towards the modes of the dataset  $X_o$  (with kernel size  $\sigma$ ) where  $F(x, X_o) = 0$ . Therefore this is a truly clustering algorithm.

Stopping the GMS algorithm to find the modes is very simple. Inasmuch as samples move in the direction of normalized gradients towards the modes that are fixed points of Eq. (7.33), the average distance moved by samples becomes smaller over subsequent iterations. By setting a tolerance ( $tol$ ) level on this quantity to a low value we can get the modes as well as stop GMS from running unnecessarily. Stop when

$$\frac{1}{N} \sum_{i=1}^N d_n(x_i) < tol, \quad d_n(x_i) = \|x_i(n) - x_i(n-1)\|. \quad (7.34)$$

### Case Study for Mean Shift Algorithms

The dataset in Figure 7.11a consists of a mixture of 16 Gaussians with centers spread uniformly around a circle of unit radius. Each Gaussian density has a



**Fig. 7.11.** (a) Ring of 16 Gaussians (b) with different a priori probabilities. The numbering of clusters is in the anticlockwise direction starting with center  $(1,0)$  (c) GMS modes; (d) GBMS modes. (from [260]).

spherical covariance of  $\sigma_g^2 I = 0.01 \times I$ . To include a more realistic scenario, different a priori probabilities were selected which are shown in Figure 7.11b. Using this mixture model, 1500 i.i.d. data points were generated. We selected the scale of analysis  $\sigma^2 = 0.01$  such that the estimated modes are very close to the modes of the Gaussian mixture. Note that because the dataset is a mixture of 16 Gaussians each with variance  $\sigma_g^2 = 0.01$  and spread across the unit circle, the overall variance of the data is much larger than 0.01.

Figure 7.11c, d, show the mode-finding ability of GMS and GBMS algorithms. To compare with ground truth we also plot  $2\sigma_g$  contour lines and actual centers of the Gaussian mixture. With the  $tol$  level in Eq. (7.34) set to  $10^{-6}$ , the GMS algorithm stops at the 46th iteration giving almost perfect results. On the other hand, GBMS stops at the 20th iteration already missing four modes (shown with arrows), and was the best result achievable by GBMS. In [280] there are many more applications of these algorithms.

These two algorithms have a direct information-theoretic interpretation with the descriptors we have introduced in Chapter 2, which shows the power of the tools being developed. Practically, they provide clustering algorithms that are fast, do not require a selection of the number of clusters and have neither stepsize nor local minima.

## 7.7 Graph-Theoretic Clustering with ITL

As discussed, one branch of clustering techniques utilizes graph theory to partition the data. Graph-theoretic clustering has the advantage that parametric assumptions about data distributions do not have to be made. In addition, it normally precludes the need to know in advance the number of clusters to be formed. In graph theoretic clustering, a proximity graph [160] is usually constructed. In a proximity graph each node corresponds to a data sample, which is considered a point in feature space. Between each pair of nodes an edge is formed, and the weight  $d(i, j)$  on each edge is a measure of the similarity (proximity) of the nodes  $i$  and  $j$ . Clustering now becomes the problem of partitioning the proximity graph.

A common partitioning method consists of creating a hierarchy of threshold subgraphs by eliminating the edges of decreasing weight in the proximity graph, such as the well-known single-link [306] and complete-link [182] hierarchical clustering algorithms. Other methods form clusters by breaking inconsistent arcs in the minimum spanning tree [347] of the proximity graph, or graphs constructed from limited neighborhoods [322].

Recently, a new line of research in clustering has emerged. It is based on the notion of a *graph cut*. A set of points,  $x_l$ ,  $l = 1, \dots, N$ , in an arbitrary data space can be represented as a weighted undirected graph  $\Gamma$ . Each node in the graph corresponds to a data point. The edge formed between a pair of nodes, say  $l$  and  $l'$ , is weighted according to the similarity between the corresponding data points. The edge-weight is denoted  $k_{ll'}$ . The graph cut

provides a measure of the cost of partitioning a graph  $\Gamma$  into two subgraphs  $\Gamma_1$  and  $\Gamma_2$ , and is defined as

$$Cut(\Gamma_1, \Gamma_2) = \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} k_{ij}, \quad (7.35)$$

where the index  $i = 1, \dots, N_1$ , runs over the  $N_1$  nodes of subgraph  $\Gamma_1$  and the index  $j = 1, \dots, N_2$ , runs over the  $N_2$  nodes of subgraph  $\Gamma_2$ . That is, the cut measures the weight of the edges that have to be removed in order to create the two subgraphs. Wu and Leahy [336] first proposed minimizing the cut-cost as a means for clustering and image segmentation. Shi and Malik [296] pointed out that the cut tends to produce a skewed data partition. It will in fact be minimized if one node in the graph is isolated in one group, and all the rest in the other group. They proposed the heuristically motivated *normalized cut* (NC), defined as

$$NC(\Gamma_1, \Gamma_2) = \frac{Cut(\Gamma_1, \Gamma_2)}{Assoc(\Gamma_1, \Gamma)} + \frac{Cut(\Gamma_1, \Gamma_2)}{Assoc(\Gamma_2, \Gamma)}, \quad (7.36)$$

where  $Assoc(\Gamma_m, \Gamma) = \sum_{i=1}^{N_m} \sum_{j=1}^N k_{ij}$  is the total connection from nodes in  $\Gamma_m$  to all nodes in the graph  $\Gamma$ . Shi and Malik optimized the normalized cut based on the eigenvectors of the Laplacian matrix  $L = D - K$ . Here,  $D$  is a diagonal matrix where the  $m$ th diagonal entry is given by  $d_m = \sum_{l=1}^N k_{ml}$ . The matrix  $K = [k_{ll'}]$ ,  $l = 1, \dots, N$  and  $l' = 1, \dots, N$ , is called the *affinity matrix*. Several other heuristically motivated cut normalizations have also been proposed, such as the min-max cut [75], the typical cut [111] and the BCut [286].

When the optimization is carried out based on the eigendecomposition (spectrum) of a matrix, the methods are referred to as *graph spectral clustering* methods. Graph spectral clustering methods are promising compared to traditional clustering methods. Other examples of spectral clustering methods can be found in [230]. The main problems associated with graph spectral clustering methods are the following.

1. An appropriate affinity measure (edge-weight) must be selected. Often, this corresponds to selecting the width of an exponential kernel function. There is no widely accepted procedure to select this parameter, even though it heavily affects the clustering result.
2. Furthermore, the  $(N \times N)$  matrix  $K$  needs to be stored in memory, and possibly other matrices too. In addition, a matrix eigendecomposition needs to be done. The computational complexity of computing an eigenvector of an  $(N \times N)$  matrix is on the order of  $O(N^2)$ . Hence, finding all the eigenvectors scales as  $O(N^3)$ .
3. It is a concern that the various graph spectral cost functions are based on heuristics, and lack a clear theoretical foundation.

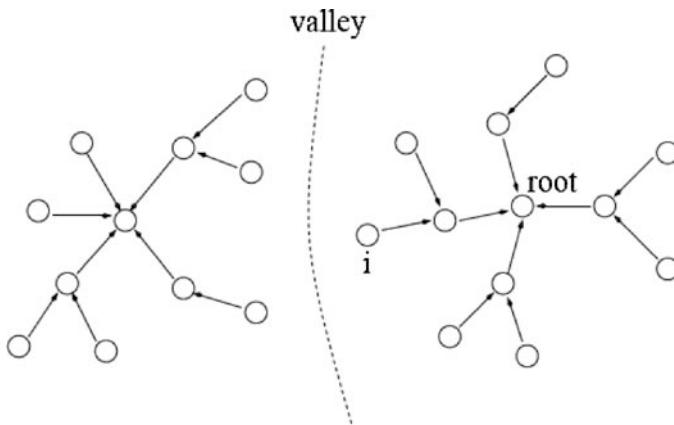
In this section we see how ITL concepts can be used for graph clustering [186]. We start by showing that it makes sense to use the information forces to cut graphs and then improve the idea using the concept of information cut [165] implemented with the fuzzy algorithm already described in Section 7.5.

### Information Forces for Graph Clustering

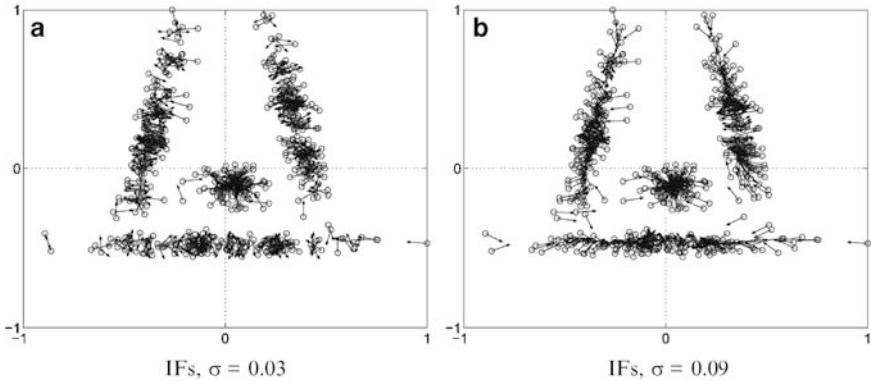
Perhaps the most interesting way to introduce ITL for graph theoretic clustering, is based on *directed trees* [165]. In a directed tree (Figure 7.12) each node  $i$  initiates a branch pointing to another node  $j$ , which is called the predecessor of  $i$ .

The root is the only node that does not have a predecessor. Starting from any node, the branches can be followed to the root. Let us assume that each node except the root has one and only one predecessor, but each could be the predecessor of a number of nodes (its “children”), including zero. The two clusters are separated by a valley, where the density of data points are low. Nodes near the valley, such as node  $i$ , must point away from the valley in order for the clusters to be formed. In [108] the predecessor  $j$  of node  $i$  is searched along the steepest ascent of the probability density function, which is estimated based on points within a local region centered around  $i$ . Node  $j$  is found within the local region, as the node closest to the steepest ascent line from  $i$ . This method is sensitive to the size of the local region, especially in the important areas near the valley.

In the ITL approach each data point can be considered as an information particle that experiences a force acting on it, pointing in the direction of a cluster. Therefore the ITL approach can be used to create directed trees



**Fig. 7.12.** Example of two directed trees, each corresponding to a cluster (from [165]).



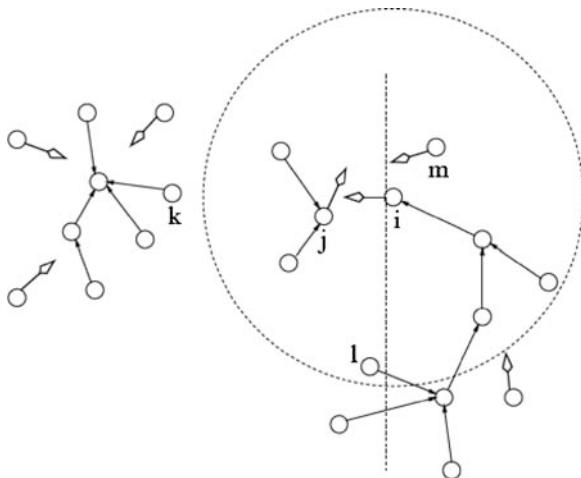
**Fig. 7.13.** (a), (b) Example of a dataset and the IFs acting on each particle for two different values of  $\sigma$ . (from [165]).

according to the direction of the information forces. The information forces (IF) contain global information about all data points, governed by the width  $\sigma$  of the kernel (e.g., Gaussian) function, so they can also be used when selecting the predecessor node  $j$ . The algorithm searches for  $j$  in a neighborhood of  $i$ , where the size of the neighborhood is specified by  $\sigma$ , such that  $j$  is closest to the direction of the information force acting on  $i$ .

Figure 7.13 shows an example of the IF for the dataset. In Figure 7.13a the IF acting on each data point is indicated by an arrow. The arrows only convey information about the directions of the forces, not the magnitude. Before calculating the IFs the dataset was normalized feature by feature to lie in a range  $[-1, 1]$ . Figures 7.13 shows the IFs for two values of  $\sigma$ . For  $\sigma = 0.03$ , it can be seen that nearly all the IFs point inward to one of the clusters. A few outliers mostly interact with each other, because the kernel size is small. For  $\sigma = 0.09$ , all the forces are pointing inward, however, the corresponding PDF estimate (not shown, see [165]) is still a rather crude and noisy estimate, indicating that if our concern is solely density estimation,  $\sigma$  is probably too low. Notice also that although there is a large difference in kernel sizes, the differences in the IFs are minor; that is, most of the forces point to the same clusters that will yield very similar clustering results.

### Creating Directed Trees

The procedure for creating directed trees is very intuitive, once the IFs have been calculated. Each data point  $x_i$  is examined, one at a time, where  $x_i$  corresponds to node  $i$  in the final tree. For node  $i$  we determine whether it has a predecessor  $j$ , or whether it is a root, based on the following. Node  $j$  is defined as the predecessor of node  $i$  if it satisfies the condition that node  $j$



**Fig. 7.14.** Predecessor selection with IF.

lies closest to the direction of the force  $F_i$  acting on  $i$  under the following constraints;

1. The distance  $\|x_i - x_j\| \leq 3\sigma$ .
2.  $F_i(x_i - x_j) \geq 0$ .
3. Node  $j$  can not be one of  $i$ 's children.

If there exists no node satisfying the above constraints, then node  $i$  is defined to be a root, not pointing to another node. The only free parameter,  $\sigma$  is the same as the one already used for the IFs. The end result of this procedure is a set of directed trees, each corresponding to a cluster.

Constraint 1 is necessary in order to avoid linking together trees that are in fact part of different clusters. Consider Figure 7.14. The tiny arrows show how nodes have been connected up to a certain iteration.

The nodes with larger arrows have not yet been examined, and the arrows show the direction of the IF acting on each one of them. Let us examine node  $i$ . Of all the nodes pointing inward to the cluster  $i$  belongs to, node  $j$  is closest to the direction of  $F_i$ . However, node  $k$ , actually belonging to a different cluster, is even closer to the direction of  $F_i$ . To avoid  $k$  being selected as the predecessor of  $i$ , we must restrict our search to a neighborhood of node  $i$ . We find that simply defining the neighborhood of  $i$  to be given by a hypersphere of radius  $3\sigma$  centered at  $i$ , is reasonable.

Our choice of neighborhood is reasonable based on the inherent properties of kernel PDF estimation. In order for the PDF estimate to be relatively accurate, the Gaussian kernel must be chosen such that its effective width is sufficiently large, but not too large, otherwise the estimate would be too smooth. The effective width of a Gaussian kernel is given by  $3\sigma$ , because it concentrates 98% of its power.

Constraint 2 is crucial, in as much as it ensures that we really use the information provided by the direction of the IFs. Figure 7.14 also illustrates this point. Node  $m$ , or any of the other nodes to the right of the dashed line, is not allowed to be selected as the predecessor of node  $i$ . The reason for this is obvious; the whole idea of our clustering technique is to use the IFs to create directed trees because the IFs point toward clusters, and not away from them.

Constraint 3 ensures that a node does not become one of its own children, contradicting the idea of a directed tree. For instance, in Figure 7.14, node  $l$  cannot be a predecessor of node  $i$ , even though it is located within the neighborhood of  $i$ .

### Case Study for Graph Clustering

Clustering experiments are presented herein, both on an artificially created dataset, and two real datasets. In all experiments the data have been normalized feature-by-feature to have a range  $[-1, 1]$ . We create the directed trees, and for each tree assign the same labels to its members. Outliers in the dataset tend to create trees with only one or a few members. Clusters with five members or less are kept in an outlier set, and are not assigned a label. Labeling these points can be done in a postclustering operation, for example, by simple nearest-neighbor classification.

First, we revisit the dataset considered in Figure 7.5 and the clustering result of the IF directed tree method produces identical clustering to the fuzzy clustering algorithm presented in Section 7.5, for a kernel  $\sigma = 0.07$ . The result is very satisfying in as much as there is no need for kernel annealing during the training. Only three patterns have been assigned to the wrong cluster, and there is only one outlier. Furthermore, the outlier would be assigned to the correct cluster after a nearest-neighbor classification. However, we have experimentally verified that even though the IFs point inward to clusters for at least  $0.03 < \sigma < 0.09$ , the overall clustering procedure is not uniformly good for this range of values, so the actual value of  $\sigma$  matters.

Next, the method is tested on the Wine dataset, extracted from the UCI repository database. This dataset consists of 178 instances in a 13-dimensional feature space, where the features are found by chemical analysis of three different types of wines. This dataset is included in the analysis, because it shows that this clustering method is capable of performing well in a highdimensional feature space. For  $0.29 < \sigma < 0.32$  we obtain satisfactory clustering results. The confusion matrix using  $\sigma = 0.32$  is shown in Table 7.3, where the numbers in parentheses indicate the number of instances actually belonging to each class. From the ten patterns assigned to the wrong class, they all belong to  $C_2$ , but some are assigned to  $C_1$ , and some to  $C_3$ . There are a total of 13 outliers.

We end this analysis with the well-known IRIS dataset, also extracted from the UCI repository database. In this case this clustering method is more sensitive to the kernel size than in the previous experiment. The range of  $\sigma$  for

**Table 7.3.** Confusion Matrix for Wine Data  $\sigma = 0.32$ 

		Result		
		$C_1$	$C_2$	$C_3$
True	$C_1$	(59)	59	0
	$C_2$	(71)	4	50
	$C_3$	(48)	0	46

**Table 7.4.** Confusion Matrix for Iris Data,  $\sigma = 0.095$ 

		Result		
		$C_1$	$C_2$	$C_3$
True	$C_1$	(50)	49	0
	$C_2$	(50)	0	42
	$C_3$	(50)	0	5

which we obtain reasonable results is narrow. However, choosing  $\sigma = 0.095$ , we obtain the confusion matrix shown in Table 7.4. In this case eight patterns are assigned to the wrong class. The incorrectly labeled patterns clearly belong to the two clusters that overlap somewhat.

The advantage of the IF for graph clustering is its ability to discover clusters of irregular shape, without having to know the true number of clusters in advance. A further advantage is the simplicity of the approach. However, the sensitivity to  $\sigma$  is a shortcoming, and it is due to the local emphasis of this algorithm.

## 7.8 Information Cut for Clustering

From the creation of the directed trees with ITL, it is apparent that the concepts of information potential fields and forces can be linked to graph theory. It turns out that the gradient descent procedure outlined in Section 7.5 to clustering based on the DCS can also be framed as an automated way to partition a graph, yielding an alternative to spectral clustering that we called the *information cut*. The information cut draws its optimality from the information-theoretic Cauchy-Schwarz divergence measure between probability density functions. The goal is to assign cluster memberships to the data points such that the information cut is minimized. Next we link the concepts of ITL to graphs such that the algorithm explained in Section 7.5 can be applied.

As we have seen in Chapter 2, the convolution theorem for Gaussian functions, states that

$$\int G_\sigma(x, x_l)G_\sigma(x, x_{l'})dx = G_{\sigma\sqrt{2}}(x_l, x_{l'}) = \kappa_{ll'}, \quad (7.37)$$

where  $G_\sigma$  is the Gaussian function. Hence the affinity matrix is built from the information potential between sample pairs. Thus, when the actual densities are replaced in the cross-information potential (which is the argument of the  $D_{CS}$ ) we obtain

$$\int \hat{p}_1(x)\hat{p}_2(x)dx = \frac{1}{N_1 N_2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \kappa_{ij}. \quad (7.38)$$

Comparing this expression with Eq. (7.35) we conclude that it is a graph cut. Hence, the samples corresponding to  $p_1(x)$  are related to the nodes of graph  $\Gamma_1$ , and the samples corresponding to  $p_2(x)$  with the nodes of graph  $\Gamma_2$ . An exactly similar calculation is done for the two quantities in the denominator of the CS distance and the new graph partitioning cost function which we call the *information cut* ( $IC$ ) is defined as

$$IC(\Gamma_1, \Gamma_2) = \frac{\sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \kappa_{ij}}{\sqrt{\sum_{i=1}^{N_1} \sum_{i'=1}^{N_1} \kappa_{ii'} \sum_{j=1}^{N_2} \sum_{j'=1}^{N_2} \kappa_{jj'}}}. \quad (7.39)$$

In graph theory, a quantity known as the *volume of a graph* is given by the sum of all the edge-weights in the graph; that is,  $Vol(\Gamma_1) = \sum_{i=1}^{N_1} \sum_{i'=1}^{N_1} k_{ii'}$ . Therefore, the information cut may also be written as

$$IC(\Gamma_1, \Gamma_2) = \frac{Cut(\Gamma_1, \Gamma_2)}{\sqrt{Vol(\Gamma_1)Vol(\Gamma_2)}}. \quad (7.40)$$

In order for the information cut to take a small value, there is a trade-off between a small cut value, and a large value for the product of the volumes. Hence, this derivation has introduced a theoretically well-defined normalization that will prevent the information cut from obtaining a minimum when one node is isolated from the rest. In the case of partitioning a graph into more than two subgraphs (i.e., subgraphs  $\Gamma_c$ ,  $c = 1, \dots, C$ ), we define the following multiway cut

$$IC(\Gamma_1, \dots, \Gamma_C) = \frac{Cut(\Gamma_1, \dots, \Gamma_C)}{\sqrt{\prod_{c=1}^C Vol(\Gamma_c)}}, \quad (7.41)$$

where  $Cut(\Gamma_1, \dots, \Gamma_C)$  is the sum of all the edge weights that need to be removed in order to create  $C$  subgraphs.

## Information Cut with Gradient Search

The issue now is to find a way to optimize the information cut cost function in Eq. (7.41). The gradient method described in Section 7.5 can be utilized

here without any modification. The big advantage of this alternative is that there is no need to precompute or store the full affinity matrix.

We initialize the membership vectors randomly according to a uniform distribution. Better initialization schemes may be derived, although in our experiments, this random initialization yields good results. Kernel size annealing should also be part of the procedure. We show experimentally that in our algorithm the convergence problem can to a certain degree be remedied, by allowing the size of the kernel to be annealed over an interval of values around the optimal value. As opposed to most graph-based clustering algorithms, the annealing procedure therefore has the effect that the affinity measure will not be fixed, but will start out large, and decrease towards a small value.

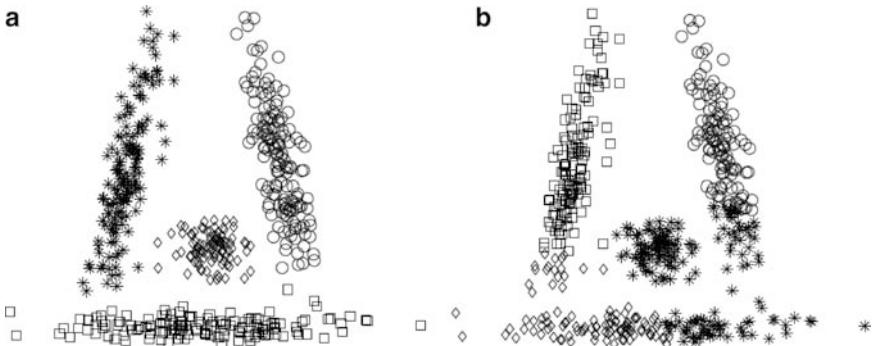
The computation of all the gradients  $\partial IC / \partial m_i$ ,  $i = 1, \dots, N$ , is done according to Eqs. (7.19) and (7.20). Because it is important to reduce the complexity of the algorithm,  $\sum_{m=1}^M m_m k_{im}$  in Eq. (7.20) is estimated by stochastically sampling the membership space and utilizing  $M$  randomly selected membership vectors. Hence, the overall complexity of the algorithm reduces to  $O(MN)$  for each iteration. We show that we obtain very good clustering results, even for very small  $M$  (e.g.  $M = 0.2N$ ). Alternatively, the incomplete Cholesky decomposition can be used to reduce the calculation and still use a matrix formulation for the algorithm.

## Case Studies with the Information Cut

This section reports some clustering experiments using the proposed information cut clustering method [165]. The number of stochastically selected membership vectors for gradient computation is determined by  $M = 0.2N$ . We compare with the normalized cut algorithm [296], which is considered by many authors to be a state-of-the-art graph-based clustering method. In [296], the normalized cut scale parameter was recommended to be on the order of 10–20% of the total range of the Euclidean distances between the feature vectors. We use 15% in our experiments.

We manually select the number of clusters to be discovered. This is of course a shortcoming compared to a fully automatic clustering procedure, but it is commonly the case in most graph-based clustering algorithms. We also normalize the variance in each feature vector dimension to one in all experiments (for both algorithms) to avoid problems in case the data scales are significantly different for each feature (both methods assume a spherical affinity measure).

The dataset of Figure 7.5 is used in this example. We provide the number of clusters,  $C = 4$ , as an input parameter to both algorithms. The Parzen window size used in the information cut algorithm is determined to be  $\sigma = 0.12$  by AMISE (Eq. (2.56)). This means that the effective kernel size used to calculate affinities between data points (nodes) is equal to  $\sigma = 0.17$ . Figure 7.15a shows a typical result obtained by the information cut algorithm. By visual

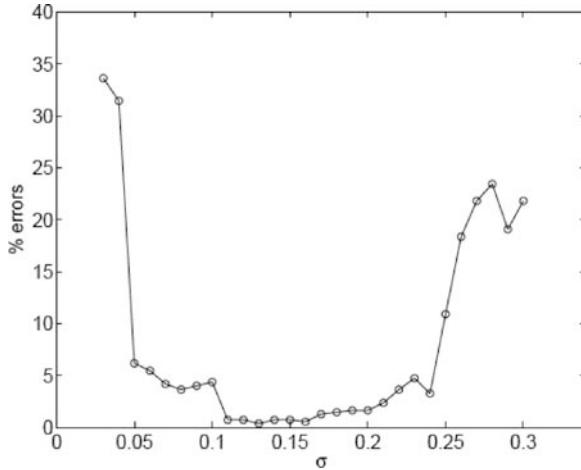


**Fig. 7.15.** Typical (70% of the trials) information cut clustering result shown in (a). Typical normalized cut clustering result shown in (b) (from [165]).

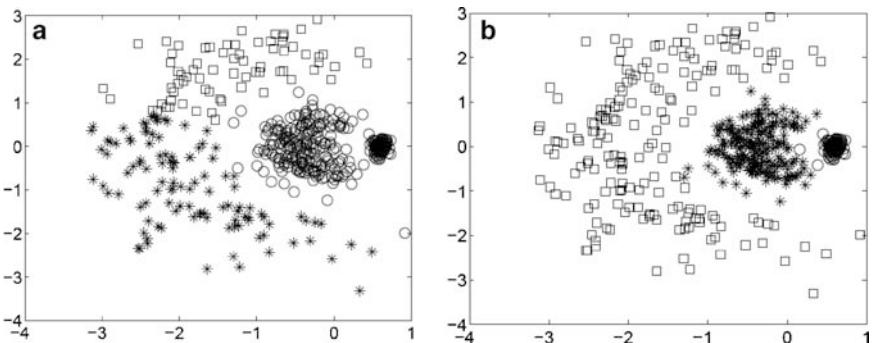
inspection, it clearly makes sense, meaning that the cluster structure underlying the dataset seems to have been discovered. Figure 7.15b shows a typical result obtained by the normalized cut algorithm. It is significantly different from the results obtained with the information cut algorithm, and seems to fail to discover the cluster structure. For this particular dataset, the scale parameter used in the normalized cut algorithm is determined to be  $\sigma_{NC} = 0.18$ . This means that the edge-weights (affinities) between nodes are roughly the same for both methods. Hence, the significantly different clustering results observed must be a consequence of (1) different clustering cost functions, (2) different optimization techniques, or both.

The next experiment illustrates the effect of the Parzen window size on the clustering results, still using the dataset shown in Figure 7.5. Figure 7.16 shows the error percentage using the information cut algorithm for a range of kernel sizes. The plot is created by running the algorithm three times for each kernel size, and then picking the best of these trials based on the value of the information cut (operating in fixed kernel mode). For  $\sigma < 0.05$ , the error percentage is very high. In the range  $0.11 < \sigma < 0.16$  the error percentage is very low, before it rises again. Note that the kernel size determined by AMISE is in the middle of this range. In the range  $\sigma > 0.25$ , the error percentage is very high again. On average, the algorithm stops after about 25 iterations.

The annealing procedure comes with a positive side-effect when it comes to coping with clusters of significantly different data scales. In Figure 7.17a, we show the result obtained by the normalized cut algorithm on a data set consisting of three clusters of different data scales. The normalized cut algorithm uses a fixed kernel size to determine node affinities, and the clustering result clearly suffers from this property. In fact, the normalized cut algorithm did not obtain satisfying results, even when manually tuning the kernel size. The result obtained by the information cut algorithm in annealing mode is shown in Figure 7.17b. All three clusters have been revealed, even though they have significantly different scales, and are separated by highly nonlinear cluster boundaries.



**Fig. 7.16.** Clustering result (error percentage compared to the generated model) obtained by the information cut algorithm over a range of kernel sizes. The “optimal” kernel size,  $\sigma = 0.12$  by Eq. (7.27), lies in the middle of the range corresponding to a low error percentage (from [165]).



**Fig. 7.17.** Comparison between the normalized cut (a) and the information cut (b) (from [165]).

*Pendigits Dataset:* This dataset was created for pen-based handwritten digit recognition (16 dimensions), and is extracted from the UCI repository. All attributes are integers in the range [0, 100]. From the test data, we extract the data vectors corresponding to the digits 0, 1 and 2. These classes consist of 363, 364, and 364 data patterns, respectively. We specify  $C = 3$  as an input parameter to the algorithm.

The clustering results are compared to the known data labels (but unknown to the algorithm). The normalized cut algorithm obtains 73.4% correct clustering on this dataset. The information cut kernel size is

**Table 7.5.** Effect of  $M$  in Performance (From [165])

$M$	$0.1N$	$0.2N$	$0.3N$	$0.4N$	$0.5N$	$0.6N$	$0.7N$	$0.8N$	$0.9N$	$N$
%	83.5	84.4	84.7	85.3	84.1	84.5	83.8	84.7	85.3	85

automatically determined to be  $\sigma = 0.63$ . Operating in annealing mode, using  $M = 0.2N$  samples for stochastic approximation, the information cut algorithm obtains 84.4% correct clustering, a clear improvement compared to the normalized cut method. In this data set, we investigate more closely the effect of the stochastic sampling approach. Table 7.5 shows the information cut result (best out of five trials) obtained for a range of  $M$ , and we conclude that the stochastic sampling approach is very effective in terms of computational complexity, at virtually no performance cost.

*Wine DataSet:* The information cut algorithm was also tested in the Wine dataset, extracted from the UCI repository. The normalized cut algorithm performs very well on this dataset, obtaining 96.6% correct clustering. The information cut algorithm performs equally well or better, obtaining 97.2% correct clustering, operating in annealing mode, with  $M = 0.2N$ . Moreover, we have argued that our optimization approach has benefits with respect to computational complexity and memory requirements. We have shown that our strategy for dealing with the problem of convergence to a local minimum, namely kernel annealing, may also have the positive side-effect of handling better different cluster data scales.

## 7.9 Conclusion

This chapter dealt with clustering using information-theoretic criteria. Clustering is a huge field but we concentrated on the subset of methods that perform clustering based on a dissimilarity cost function. The role of information-theoretic cost functions in clustering was developed from first principles and how it bridges very different classes of algorithms, such as the mean shift, graph cuts, and spectral clustering. These relationships are only possible due to the abstract level and the usefulness of the Cauchy-Schwarz divergence measure that can be directly estimated from data using kernels. We also would like to emphasize how the analogy to information forces is able to unravel how the algorithms cut the data, which is particularly clear in the mean shift and in the graph cut methods.

For most of these algorithms what is needed is just the cross-information potential because in optimization monotonic functions (the log) and scaling factors do not affect the location of the extreme of the cost function, which simplifies the algorithms. Once a reasonable cost function is selected, a search algorithm is still needed to search for the extreme. Here we fuzzify the class membership to be able to derive a gradient search method for clustering.

The algorithm is very efficient computationally, but it is sensitive to local minima. We demonstrated on several occasions the positive effect of using kernel annealing during the adaptation, which is able to avoid local minima with the proper annealing rate. However, this brings three extra parameters to clustering. Use of the fast Gauss transform will speed up the clustering algorithms. All the algorithms presented have performances comparable to or better than the corresponding ones available in the literature.

More important several different clustering methodologies have been coupled: mean shift was shown to be related to entropy minimization with a penalty given by the Cauchy-Schwarz divergence; graph-based clustering was coupled to non-parametric density estimation in terms of Parzen windowing. Consequently, it was observed that the Parzen window width directly determines the affinity measure used to calculate edge-weights in the graph. It is well known that it is crucial for any graph-based method to determine this parameter appropriately, but few data-driven methods exist. We used the simplest approach for data-driven kernel size selection, namely Silverman's rule. However, more advanced techniques can easily be incorporated. We also proposed a novel information cut for graph clustering that seems to work better than the conventional method used in the field. Moreover, we show that effectively these methods are intrinsically related to spectral clustering, having some practical advantages but also some disadvantages (requiring the selection of the number of clusters before hand). We believe that much more interesting work can be done using the ITL concepts for clustering, and really show the fundamental role that information-theoretic methods have for clustering.

## Self-Organizing ITL Principles for Unsupervised Learning

Sudhir Rao, Deniz Erdogmus, Dongxin Xu, and Kenneth Hild II

### 8.1 Introduction

Chapter 1 presented a synopsis of information theory to understand its foundations and how it affected the field of communication systems. In a nutshell, mutual information characterizes the fundamental compromise of maximum rate for error-free information transmission (the channel capacity theorem) as well as the minimal information that needs to be sent for a given distortion (the rate distortion theorem). In essence given the statistical knowledge of the data and these theorems the optimal communication system emerges, or self-organizes from the data.

One of the fundamental goals of learning is exactly to learn the statistics from the data, and to self-organize or adapt in such a way as to guarantee optimality in some sense. We saw in Chapter 7 how ITL could be successfully used for clustering, but it is not straightforward to extend the same principles for more general forms of self-organization. Indeed, unsupervised learning is normally thought of as a strictly data-driven learning process exploiting local structure and similarity within the data, such as the k-means and ITL clustering algorithms.

How biological organisms interact with their environments has naturally been a source of inspiration, because they are a living proof that it is possible to extract useful information from a time-varying and unknown environment for goal-driven behavior using the external data and the genetic code, that is, mostly in an unsupervised fashion. In this sense biological organisms are the ultimate learning machines, but the principles at play are far from being well understood, and naturally have attracted lots of attention from the learning community.

Information-theoretic descriptors quantify the essence of the communication process assuming full statistical knowledge of the data, but the real question is how to manipulate them to create models of the world when the statistics of the task are unknown. What are the optimal strategies for self-organization using information theoretic principles? Are the channel capacity

and the rate distortion theorems still the holy grail of optimal design for learning, or are there any other information principles yet to be discovered? These are some of the interesting questions that are only partially answered, and that are addressed in this chapter.

Information theory has solid algorithmic procedures to manipulate entropy and mutual information using global structure, namely the MaxEnt and the MinXEnt principles that were proposed in the physics and statistics literatures [157, 161]. These principles were developed to create distributions that would match the explicit cost constraints given the data, and they are the starting point for the chapter. However, there are other unsupervised algorithms that mimic supervised learning in the sense that they require an adaptive system with adaptable parameters optimized from a global cost function, which have been called energy-based algorithms [196]. Again here information-theoretic costs play a significant role and they give rise to what has been called self-organized information-theoretic principles [152]. We present the case of Linsker's information maximization [199], Becker and Hinton Imax [24], and Barlow's principle of entropy minimization [24] that are a direct application of information theory ideas derived for communications, dressed now as cost function for adaptation. But the learning problem is richer than optimizing a communication channel, therefore we can expect that other ITL-based principles are waiting to be discovered. For instance, when Bell and Sejnowski coupled ITL with nonlinear adaptive systems, a very powerful new way of performing independent component analysis surfaced [316].

As mentioned in Section 1.6, we further believe that the true advantage of self-organizing information theoretic principles when compared to other local unsupervised learning rules is to focus on the optimality of the cost function. We already saw how productive this line of reasoning can be for entropic and divergence cost functions, with the concepts of information potential and information forces that link the global cost, no matter if supervised or unsupervised, to the rules of the interactions everywhere in the space. In an analogy with physics, each physical law describes interaction behaviors that lead to quantifiable optimality. However, they can be utilized in a wide range of scenarios, and foster very different behavior depending upon the embodiment.

To illustrate this more abstract view of unsupervised learning two other self-organizing principles are discussed: Tishby et al, information bottleneck (IB) method [316] that for the first time brings explicit purpose into the optimization framework; and a related principle recently proposed by our group called the *principle of relevant information* (PRI) [259] which brings a coherent view to unsupervised learning algorithms that were originally developed specifically for different application (e.g., clustering, principal curves, or vector quantization). We end the chapter with examples of how Renyi's entropy and the information potential can be applied in practice to implement all the self-organizing principles, without having to constrain the data PDF nor the linearity of the adaptive system as done in the past to obtain analytical solutions.

## 8.2 Entropy and Cross-Entropy Optimization

In many experimental conditions, the collected data are the only source of knowledge about the problem, which in a Bayesian framework, raises the question of the prior probability selection. Any function that obeys the definition of a PDF can be a possible prior i.e.,  $\sum_{i=1}^N p_i = 1$ ,  $p_i \geq 0$ . Laplace formulated a principle now called the *principle of insufficient reason* that basically stated “... in the solution of a problem the researcher should only use what is known about the problem and nothing else.” Laplace argued that when there is no reason to believe that a stochastic event will occur preferentially compared to other outcomes, the most reasonable solution is a uniform distribution for the probabilities of occurrences.

Information theory (IT) provides a powerful inference methodology to describe general properties of arbitrary systems on the basis of scarce measurements, hence its appeal to machine learning. Jaynes proposed in 1958 the maximum entropy principle (MaxEnt) which basically states that the experimenter should choose a distribution that maximizes the entropy of the data given the constraints (in fact agreeing with Laplace) [161]. Inasmuch as entropy measures uncertainty in the data, maximizing entropy should add the least of our a priori knowledge of the problem into the solution, and let the data “speak for themselves.”

Moments of the distribution are viable alternatives to implement the constraints or in general we can constrain the expected values of functions  $g_i(X)$ ,  $\sum_{i=1}^N p_i g_k(x_i) = a_k$ ,  $k = 1, 2, \dots, m$  where the random variable  $X$  take values  $x_1, \dots, x_N$  with corresponding probabilities  $p_1, \dots, p_N$ . In this way a set of  $m + 1$  constraints on the mass function are created (the extra constraint is that the sum of  $p_i$  must be one) but if  $m + 1 < N$  then there will be many possible solutions. A much better strategy is to use the method of Lagrange multipliers and find the probability mass function (the  $p_i$ ) that obey all the available  $m + 1$  constraints when entropy is maximized. Therefore one should

$$\begin{aligned} \max_p H = & - \sum_{i=1}^N p_i \ln p_i, \quad \text{subject to } \sum_{i=1}^N p_i g_k(x_i) = a_k, \\ & k = 1, 2, \dots, m, \quad \sum_{i=1}^N p_i = 1 \end{aligned} \tag{8.1}$$

for  $1 \geq p_i \geq 0$ ,  $i = 1, \dots, N$ . The Lagrangian is given by

$$L = - \sum_{i=1}^N p_i \ln p_i - (\lambda_0 - 1) \left( \sum_{i=1}^N p_i - 1 \right) - \sum_{k=1}^m \lambda_k \left( \sum_{i=1}^N p_i g_k(x_i) - a_k \right), \tag{8.2}$$

Now taking the derivative of  $L$  with respect to the unknown  $p_i$  yields

$$\frac{\partial L}{\partial p_i} = 0 \rightarrow -\ln p_i - \lambda_0 - \sum_{k=1}^m \lambda_k g_k(x_i) = 0,$$

which shows that the maximum entropy distribution given any possible constraints involving linear combinations of  $p_i$  is always an exponential

$$p_i = \exp(-\lambda_0 - \lambda_1 g_1(x_i) - \cdots - \lambda_m g_m(x_i)) \quad (8.3)$$

The multipliers can be determined by substituting back  $p_i$ , yielding

$$a_k = \frac{\sum_{i=1}^N g_k(x_i) \exp\left(-\sum_{j=1}^m \lambda_j g_j(x_i)\right)}{\sum_{i=1}^N \exp\left(-\sum_{j=1}^m \lambda_j g_j(x_i)\right)} \quad k = 1, 2, \dots, m \quad (8.4)$$

The Lagrange multipliers are the partial derivatives of entropy at its maximum value w.r.t. the  $a_k$ , and the maximum value is  $H_{\max} = \lambda_0 + \sum_{i=1}^m \lambda_i a_i$ .

This result can also be applied to the continuous variable case only with trivial modifications (changing summations to integrals). The maximum entropy distribution for the range  $[-\infty, \infty]$  when the mean and the variance are constrained is the Gaussian, whereas it becomes the uniform PMF for finite range.

Jaynes showed that the three most common distributions in statistical mechanics can be obtained using the MaxEnt principle applied to different physical constraints on the energy of particles: Maxwell–Boltzmann distributions when the expected energy of the particles is constrained; the Bose–Einstein distribution when the number of particles in the system is also known (besides the mean energy); and the Fermi–Dirac distribution when the energy states can contain either one or zero particles. In signal processing the Burg spectral estimator was derived using the MaxEnt principle, and it has also been applied in independent component analysis. Above all, MaxEnt is a tool to optimize entropy and is therefore very relevant for information theoretic learning.

## Minimum Cross-Entropy Optimization

Another information optimization procedure is based on the concept of Kullback–Leibler (KL) divergence. The KL divergence is defined for two PMFs  $p$  and  $q$  as  $D_{KL}(p||q) = \sum_{i=1}^N p_i \ln(p_i/q_i)$ . The divergence can be used much like the entropy to create a different optimization principle, but now we have two PMFs so one can set the optimization problem in several ways. Generalizing the MaxEnt principle it makes sense to select the distribution  $q$  for the

comparison (representing some a priori knowledge about the solution) and finding a distribution  $p$  that satisfies the data constraints and is the closest to  $q$ . We call this the MinKL principle. It turns out that there is an interesting relationship between MaxEnt and MinKL: The solution of MaxEnt coincides (for discrete random variables) to the solution of MinKL when the given distribution is the uniform and the constraints remain the same. The MinKL problem is formulated as follows,

$$\min_p D_{KL}(p||q), \quad \text{subject to } \sum_{i=1}^N p_i g_k(x_i) = a_k \quad k = 1, 2, \dots, m \quad \sum_{i=1}^N p_i = 1 \quad (8.5)$$

and the solution becomes (the derivation is exactly the same as MaxEnt)

$$p_i = q_i \exp(-\lambda_0 - \lambda_1 g_1(x_i) - \dots - \lambda_m g_m(x_i))$$

which coincides with the MaxEnt except for the multiplication with  $q_i$ . Likewise we can obtain the constraints and Lagrange multipliers in a very similar fashion.

However, in the literature, the optimization principle based on the KL divergence is also called minimization of cross-entropy or MinxEnt. However with Eq. (8.5) the final distribution is not the cross-entropy between  $p$  and  $q$ . To obey the definition of cross-entropy, the problem must start by selecting the distribution  $p$  and finding the distribution  $q$  that satisfies the constraints. Obviously, the solutions are different. The MinxEnt problem can be formulated as

$$\min_q D_{KL}(p||q), \quad \text{subject to } \sum_{i=1}^N q_i g_k(x_i) = a_k \quad k = 1, 2, \dots, m \quad \sum_{i=1}^N q_i = 1 \quad (8.6)$$

and the solution becomes

$$\begin{aligned} q^* &= \arg \min_q \sum_{i=1}^N p_i \ln \frac{p_i}{q_i} = \arg \min_q - \sum_{i=1}^N p_i \ln q_i \\ L &= - \sum_{i=1}^N p_i \ln q_i - \lambda_0 \left( \sum_{i=1}^N q_i - 1 \right) - \sum_{k=1}^m \lambda_k \left( \sum_{i=1}^N q_i g_k(x_i) - a_k \right) \\ \frac{dL}{dq_i} &= -\frac{p_i}{q_i} - \lambda_0 - \sum_{k=1}^m \lambda_k g_k(x_i) = 0 \\ q_i &= \frac{p_i}{\left( -\lambda_0 - \sum_{k=1}^m \lambda_k g_k(x_i) \right)} \end{aligned}$$

The concavity of the logarithm endows Shannon entropy of many important mathematical properties that have been crucial to define the generalized forms of entropy reviewed above, and that yield some of its desirable features.

For information-theoretic studies, these two mathematical properties are very useful:

- When the stationary value of Shannon entropy subject to linear constraints is found, the global maximum is always obtained (no need for further checks).
- Maximizing Shannon entropy subject to linear constraints using the Lagrange method always yields quantities that are greater than zero. This is particularly useful because in constrained optimization imposing the nonnegativity constraints is in general difficult, and here it is unnecessary.

### 8.3 The Information Maximization Principle

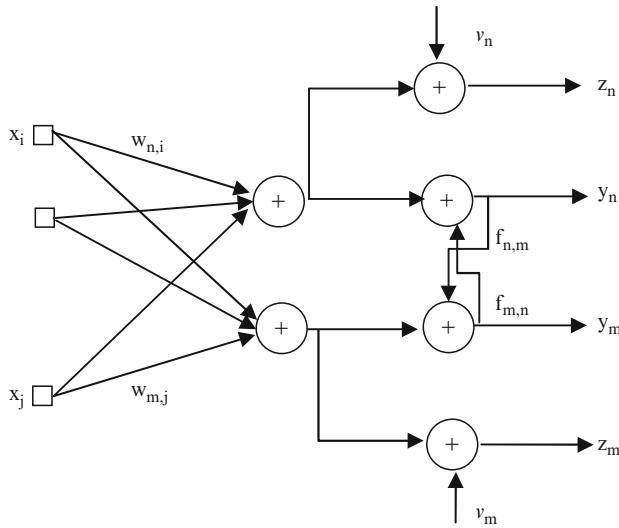
Ralph Linsker in a series of papers [199] enunciated a very general learning principle in distributed networks called the *information maximization principle* (InfoMax). He stated that to self-organize distributed systems with multiple layers (deep networks) the early processing layers should transfer as much information as possible about the input to the subsequent layers. Formally, Linsker proposed maximizing the mutual information between the output and the signal portion of the input (assumed corrupted by noise). To be biologically plausible, the optimization should be expressed by local interaction rules, otherwise the learning complexity would be factorial and would therefore be unrealistic in neural systems. The achievement of Linsker's work is that he was able to show that for linear networks and multivariate Gaussian signal and noise distributions, the global mutual information cost function for a distributed network can be implemented locally by the simple Hebbian learning rule, which is an unexpected result.

#### Linsker's InfoMax Principle

The problem setting is the following. Let us assume that the input to a linear feedforward network possibly followed by a nonlinear element that does not affect the training (Widrow's Adaline), is a random variable  $X = \{x_i\}$ ,  $i = 1, \dots, N$ , which is composed either by the signal portion  $X = S$  alone or by  $S$  disturbed with noise  $N$ , that is,  $X = S + N$  both drawn from zero-mean multivariate Gaussian distributions. The network's output is  $z = y + \nu$  where  $y = \mathbf{w}^T \mathbf{x}$  and the observation noise  $\nu$  is assumed zero-mean Gaussian and independent of  $Y$ . See Figure 8.1.

The information that  $Z$  conveys about  $S$  is calculated by  $I(Z, S) = H(Z) - H(Z|S)$ . For Gaussian distributed signal and noise, these quantities are easily computed analytically because Shannon's entropy for the multivariate Gaussian is (apart from constants)  $H(X) = 1/2 \ln(\det \Sigma)$  where  $\Sigma$  is the covariance of  $X$ . Therefore, for our case

$$I(Z, S) = 1/2 \ln(\det \Sigma_S) - 1/2 \ln(\det \Sigma_N), \quad (8.7)$$



**Fig. 8.1.** Linsker's lateral connection network.

where

$$\begin{aligned}\Sigma_S &= \mathbf{w} q_S \mathbf{w}^T + r \\ \Sigma_N &= \mathbf{w} q_N \mathbf{w}^T + r \\ q_S &= E[\mathbf{S} \mathbf{S}^T] + E[\mathbf{N} \mathbf{N}^T] \\ q_N &= E[\mathbf{N} \mathbf{N}^T] \\ r &= E[\mathbf{v} \mathbf{v}^T]\end{aligned}$$

The subscripts  $S$  and  $N$  refer to the fact that the entropy of the output  $H(Z)$  depends on  $S + N$  and  $H(Z|S)$  depend on  $N$  only, but they have the same basic form due to the Gaussian assumption on  $S$  and  $N$ . In order to derive the learning rule to adapt the network weights, assume that the noise  $\mathbf{v}$  is independent of the system weights  $\mathbf{w}$ , so the gradient has the same form for both terms (see Eq. (8.7))

$$\frac{\partial H}{\partial w_{ij}} = (\Sigma^{-1} \mathbf{w} q)_{ij}. \quad (8.8)$$

This gradient can be interpreted as an Hebbian update because of the following reasoning. In a linear network such as the one assumed here,  $w_{ij}$  would be the product of the output  $y_i$  times  $x_j$ ;

$$\Delta w_{ij} = \eta \frac{\partial H}{\partial w_{ij}} = \eta E[Y_i X_j]. \quad (8.9)$$

Comparing Eq. (8.9) with Eq. (8.8)  $q_j$  yields  $x_j$  directly but  $y_i$  is equivalent to the multiplication of  $\Sigma^{-1}$  by the corresponding weight vector  $w_{ij}$ , which implies in principle nonlocal learning. However,  $\Sigma^{-1}\mathbf{w}$  can still be computed locally by enhancing the Adaline output layer with lateral connections among the output units (Fig. 8.1) as long as the lateral weights are  $\mathbf{F} = \mathbf{I} - \alpha\Sigma$  [9]. These weights stabilize over time and produce outputs  $Y_i(\infty)$  when the learning rate is appropriately selected. Therefore, the final rule becomes  $\Delta w_{ij} = \eta E[\alpha Y_i(\infty) X_j]$  [199], where the familiar Hebbian update is noticeable, and  $\alpha$  is the stepsize of the recursive algorithm that updates the lateral connections such that  $\alpha\mathbf{Y}(\infty) = \Sigma^{-1}\mathbf{Y}$ . This means that Infomax, in a linear network with Gaussian inputs, defaults to a manipulation (maximization) of output variance terms.

According to Linsker [199], a network that learns how to maximize  $I(Z, S)$  must learn how to distinguish input signal from noise, which can be accomplished by two-phase learning. In the learning phase examples of  $S + N$  are shown, followed up by an unlearning phase where examples of  $N$  alone are provided. Therefore the total weight update will be a sum of the Hebbian term with an anti-Hebbian term produced by the unlearning phase as

$$\Delta w_{ij} = \eta(E[\alpha_S Y_i(\infty; S) X_j] - E[\alpha_N Y_i(\infty; N) X_j]). \quad (8.10)$$

Equation (8.10) has a very nice interpretation. In fact, maximizing mutual information between the input and output of a linear system with Gaussian noisy inputs corresponds to maximize output variance when the  $S + N$  input is applied and to minimize output variance when only the noise is applied, or if we prefer, to use a Hebbian update when  $S + N$  is applied and anti-Hebbian update when the noise only is applied. Therefore the network can still use second-order statistics and local adaptive rules to estimate entropies provided the signal and noise distributions are Gaussian distributed (which is, however, an unrealistic limitation).

This reasoning also means that information-theoretic principles are really unnecessary in the Gaussian and linear scenario, because under these assumptions maximum likelihood would have been sufficient to reach the same conclusion (output variance maximization). So in our opinion, the real benefit of information-theoretic reasoning is obtained when the linearity and Gaussian assumptions are lifted as remarked several times in the previous three chapters. The other notable aspect is that the weights adapted by Eq. (8.10) will grow without bound for a linear system, as is expected from the Hebbian update [233]. The nonlinearity of the Adaline output PEs constrains the output and stabilizes learning, or alternatively one could utilize Oja's rule, which normalizes at each iteration the weights by their norms [233]. Quadratic mutual information explained in Chapter 2 and used in Chapter 6 can directly implement Linsker's Infomax principle without imposing the constraint of linear networks and Gaussian inputs if applied to a network with output lateral connections to make the learning rule local to the weights. Therefore the information potential and forces help us explore Infomax in more realistic settings.

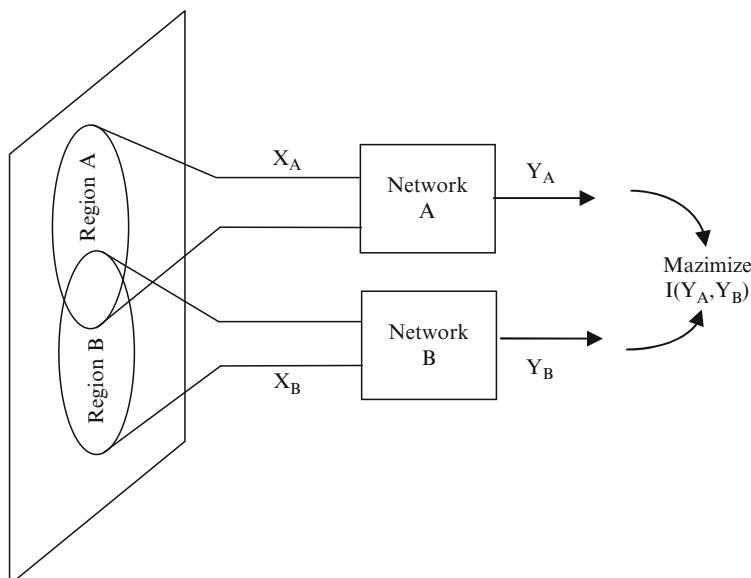
## 8.4 Exploiting Spatial Structure for Self-Organization

As we saw in Section 8.2 the InfoMax principle requires a “supervisor” to determine if the input is  $S + N$  or only  $N$ , which may be unrealistic. Internally derived teaching signals can be derived if one exploits spatial information of the input as proposed by Becker and Hinton in their I-Max principle [24]. If one assumes that the external world is structured, neighboring parts of the sensory input will have common causes in the world. If the processing system is modular with sensory inputs that overlap somewhat, a possible self-organization principle will impose that the outputs of neighboring local networks produce outputs that agree to a certain extent (Fig 8.2). The problem is that if one attempt to use the MSE between neighboring module outputs as a cost to adapt the module parameters, similar outputs among modules are obtained that discard the input structure.

Becker and Hinton showed experimentally that a good measure of agreement for modules A and B with scalar outputs is obtained by

$$\frac{1}{2} \log \frac{\Sigma(y_a + y_b)}{\Sigma(y_a - y_b)}, \quad (8.11)$$

where  $\Sigma$  is the variance obtained over the training set. This result can be interpreted in two different ways: first, it can be interpreted in a linear Gaussian setting as a cost function for canonical correlation, which finds linear combinations  $Y_A = w_A^T X_A$  and  $Y_B = w_B^T X_B$  to maximize correlation between



**Fig. 8.2.** Exploiting spatial coherence in local models with mutual information.

them. Second, it turns out that if we assume that the input to the modules accepts a Gaussian model, Eq. (8.11) is nothing but the mutual information between the signal and the average of  $Y_A$  and  $Y_B$ ; that is,

$$I(Y_A, Y_B) = -\frac{1}{2} \log(1 - \rho_{AB}^2), \quad (8.12)$$

where  $\rho$  is the correlation coefficient between the signals  $Y_A$  and  $Y_B$ , which is equivalent to Eq. (8.11). If we assume that  $X$  is again  $S + N$  as in Section 8.2, then the external control to recognize when the signal is present is no longer necessary (we assume that  $N$  is spatially uncorrelated but the signal has spatial structure).

Therefore I-max is an information-theoretical principle similar to InfoMax but now applied to spatial coherence: the transformation of a pair of vectors  $X_A$  and  $X_B$  should be chosen to maximize the mutual information between the pair of scalar outputs  $Y_A$  and  $Y_B$ . Maximizing output mutual information among the modules trains them to preserve the underlying structure present in the input scene.

Becker has been producing impressive results with this simple concept, first applied to stereoscopic vision [25]. She started with a very simple linear model for coherence at the pixel level, and later generalized to mixture models that even allow for different type of coherence in the scenes. Quadratic mutual information (QMI<sub>CS</sub>) in Chapter 2 can achieve this maximization directly without requiring the Gaussian assumption. Because this is equivalent to a generalization of canonical correlation, there are many signal processing problems that can benefit from the ITL formulation.

## 8.5 Principle of Redundancy Reduction

Another area of great interest in perception is to understand the role of redundancy in sensory coding. Attneave [10] and Barlow [16] hypothesized that redundancy plays a very fundamental role in characterizing the environment and therefore should be part of biologically plausible learning models. This led Barlow to enunciate a self-organizing principle for sensory processing as follows: The purpose of early visual processing is to transform the highly redundant sensory input into a more efficient factorial code, which makes the neural outputs mutually statistical independent given the input.

When the brain receives a combination of stimuli that form a percept, it needs to know the probability of occurrence of the percept for proper interpretation in a Bayesian framework. Biological systems are keen on recognizing unusual stimuli that may carry information, therefore they must have found a way to assign a priori probabilities to events. Due to the huge dimensionality of the space, it is unlikely that this is achieved through the quantification of the joint probability density of stimuli and rewards which grows supralinear with stimulus dimension. However, if the space of the sensory variables is

sufficiently sparse (i.e., if all sensory variables are made mutually statistically independent), then the joint probability of any two or more sensory variables can be determined knowing only the marginal probabilities of each. Then it is possible to assign probabilities to any logical function of the event from the individual sensory variable probabilities without further knowledge. Minimum entropy encoding measures the entropy of each individual event and chooses the reversible code for which the sum of entropies is at a minimum [17].

When perception is set up in this manner, the relevant steps are (1) estimation of entropies from sensory data and (2) finding the code with the minimum sum of marginal (bit) entropies. Let's assume that we have a representative ensemble of sensory data  $A$ , where each "event" appears with probably  $A_i$ , so  $H(A) = -\sum_i A_i \log(A_i)$ . The estimation of entropy from the sensory data can be alternatively accomplished with the information potential of Chapter 2. The problem of minimum entropy coding is a bit more complex. The sum of marginal (bit) entropies for a binary alphabet of symbols  $A$  that occur with probabilities  $A_i$  with a code  $c$  is given by

$$H(A, c) = - \sum_i p_i \log(p_i) - \sum_i q_i \log(q_i), \quad (8.13)$$

where  $p_i = 1 - q_i$  is the probability of the  $i$ th bit in the code to take the value 1 (and  $q_i$  is the probability of the bit to take the value 0). We know that  $H(A, c)$  is always larger than or equal to  $H(A)$  with equality for factorial codes [330]. A code is factorial if  $A_j = \pi_j$ , with  $\pi_j = \prod_{c_{ij}=1} p_i \prod_{c_{ij}=0} q_i$ , that is, when the joint density of the symbols in the code words factorizes, where  $c_{ij}$  denotes the  $i$ th bit of the code for the  $j$ th event. The problem is that finding factorial codes requires an exhaustive search over the alphabet. One simple and quite good approximation is the binary sequence coding provided the probabilities of the input alphabet form a geometric progression that can be implemented as follows. Compute the probability of the events in the alphabet, order them in descending order of probability, and assign the code words with an increasing number of ones to the list. Just by looking at the number of ones in the code, we can infer the probability of the events that originated them, so unusual activity can be pinpointed easily. The solution of the minimization problem can be pursued with autoencoders as used in population codes by Zemel and Hinton [348], or alternatively ITL cost functions if one frames finding minimum entropy codes as mutual information minimization between the events and their codes (i.e.,  $H(A, c) - H(A) = -\sum_j A_j \log(\pi_j/A_j)$ ).

Barlow's work stressed the importance of sparseness, the simplification operated by independence in extracting information from high-dimensional data and the role of entropic concepts in sensory coding. Later on, Atick and Redlich postulated the principle of minimum redundancy for the visual system following these ideas [9]. These principles transcend computational neuroscience application and are applicable in general engineering system design.

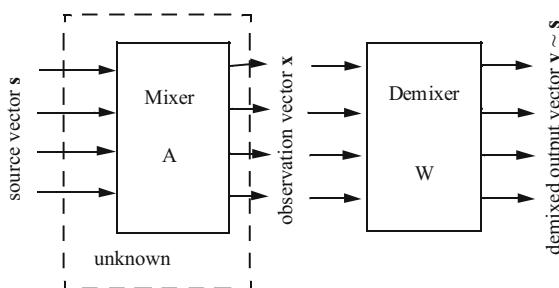
## 8.6 Independent Component Analysis (ICA)

We have seen in the above principles that statistical independence is a simplifying factor for a broad class of data analysis scenarios. The fundamental problem is how to achieve it in practical conditions using simply the data and local learning rules. The importance of Bell and Sejnowski's work [26] was exactly to show that a simple local learning rule applied to a nonlinear network (where the nonlinearity was matched to the cumulative distribution function (CDF) of the inputs) trained to maximize the entropy at its output was able to accomplish this seemingly difficult task. Independent component analysis (ICA) was already being applied in signal processing using higher-order statistical manipulations [62], but the elegance of the Bell and Sejnowski formulation was fundamental to expand the field, as we explain next.

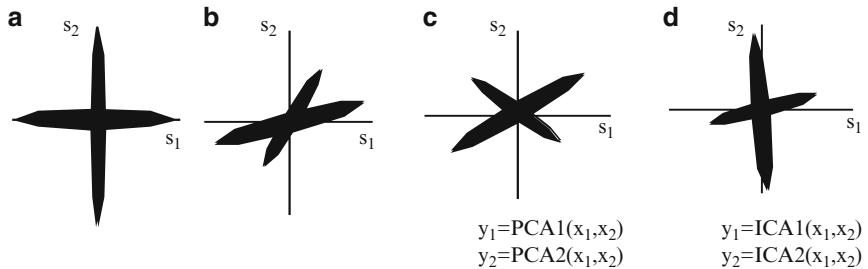
A typical instantaneous linear mixture problem that benefits from an ICA solution consists of  $N$  observations that are linear combinations of  $M$  mutually independent source signals ( $N \geq M$ ) (Fig. 8.3). The observed data  $X = AS$  is a linear nonsingular mixture ( $\dim A = M \times M$ ) of source signals  $S = (s_1, \dots, s_M)^T$ , independent of each other. There is no further information about the sources nor mixing matrix, hence the name blind source separation (BSS). The problem is to find a linear projector  $W$  such that  $Y = WX$  ( $\dim W = M \times M$ ), and  $Y = S$  apart from a permutation and scaling.

In the BSS literature, the square mixture case where the number of measurements is equal to the number of sources is the most investigated, because if there are more sources than measurements, the source recovery cannot be achieved without large distortions, although the mixing matrix can still be identified in some cases.

It is important to understand why second-order statistics cannot solve this problem. Let us assume that there are two independent sources that are mixed by an unknown matrix  $A$  (Figure 8.4a). A matrix rotates and scales the source input vector, therefore in the space of the mixtures there is a rotation and scaling of the sources (Figure 8.4b). Now if PCA is applied to the data in observation space and the principal directions are used as the projection directions for the demixer, some separation may be achieved but the result is



**Fig. 8.3.** The source separation problem.



**Fig. 8.4.** How to achieve demixing in the output space. (a) independent sources; (b) mixed sources; (c) demixed sources by PCA; (d) demixed sources by ICA.

far from optimal (Figure 8.4c). One needs to find a way to rotate the sources such that they coincide again with the axis of the output space (Figure 8.4d).

Principal components analysis (PCA) is a tool that investigates the second-order statistical structure underlying random vectors. Basically, the purpose of PCA is to find a set of orthogonal directions where the data have maximal variance [233], therefore the projections are uncorrelated with each other. Independent component analysis strengthens uncorrelatedness to independency of the projected random variables [155]. Besides its theoretical appeal in factor analysis [62], ICA is interesting as a design principle in many signal processing problems.

As a final remark, the BSS problem is broader in scope than ICA because it seeks to separate a set of unknown source signals that are mixed by an unknown mixture using either (or both) statistical independence as in ICA or decorrelation over the lags of the time signals [335]. In this book, we only consider the ICA model for the BSS problem, which requires an independent assumption amongst the sources (at least approximately).

### Bell and Sejnowski Algorithm

By definition (see Chapter 1), statistical independence among the components of a multidimensional random variable is obtained when the joint is equal to the product of the marginal densities, which can be evaluated by Shannon's mutual information (when it yields the value zero). However, there are simplifying factors when applying this idea in ICA. In fact, the mutual information of a multidimensional input – output system can also be written in terms of the entropy of the output minus the conditional entropy of the output given the input. When the goal is to adapt the system weights, it turns out that the conditional density can be dropped from consideration because it is independent of the system parameters [228]. Therefore, independence can be achieved by maximizing simply the system's output joint entropy. But this argument does not clearly state how to accomplish the task with local adaptation rules.

Bell and Sejnowski's insight was fundamental to solve this problem. Let us assume first that the system is single-input – single-output with sigmoidal

nonlinear (logistic) output units. The logistic function is a reasonable model of the cumulative density function of the input, assuming PDF unimodality, symmetry, and super-Gaussianity. In this case, if one maximizes the output entropy making it uniform (because the logistic function has finite range), then the system output density before the nonlinearity is being implicitly matched to the statistics of the input. Therefore when the nonlinearity matches the CDF of the input data the system is transferring as much mutual information from its input to the output, as Linsker prescribes in his InfoMax principle. Moreover, the learning rule for this system is trivial. In fact, if we interpret the role of the network as a smooth transformation of its input  $y = f(x, w)$ , it is well known [235] that the system's output PDF is related to the PDF of the input by  $p(y) = p(x)/|\partial y/\partial x|$ , so we immediately obtain

$$H(y) = E [\ln |\partial y/\partial x|] - E [\ln p(x)]. \quad (8.14)$$

The second term of Eq. (8.14) can be dropped because for deterministic mappings it equates to the lowest entropy and it is independent of the system parameters, so maximizing the entropy of  $y$  by changing parameters  $w$  is equivalent to performing gradient ascent in the first term of Eq. (8.14). Using the stochastic gradient idea we can drop the expected value to yield

$$\Delta w = \eta \frac{\partial H(y)}{\partial w} \rightarrow \Delta w = \eta \frac{\partial \ln |\partial y/\partial x|}{\partial w} = \left( \frac{\partial y}{\partial x} \right)^{-1} \frac{\partial}{\partial w} \left( \frac{\partial y}{\partial x} \right). \quad (8.15)$$

In the case of the logistic sigmoid  $y = 1/(1 + e^{-u})$  with  $u = wx + w_0$  this gives

$$\begin{cases} \Delta w = \eta(1/w + x(1 - 2y)) \\ \Delta w_0 = \eta(1 - 2y). \end{cases} \quad (8.16)$$

Notice that the learning rule is anti-Hebbian because of the negative product  $xy$ , but it is “regularized” because it repels the weight from zero due to the  $1/w$  term, as well as from the extreme of the sigmoid due to  $(1 - 2y)$  in the second equation; that is, it keeps the operating point in the middle of the sigmoid range. So this rule maximizes the output entropy for the scalar case, but the real importance is for the multidimensional output case. Following a similar path, one can show that for the multidimensional input–output case, Eq. (8.16) becomes

$$\begin{cases} \Delta \mathbf{w} = \eta \left( \left[ \mathbf{W}^T \right]^{-1} + (1 - 2y) \mathbf{x}^T \right) = \eta (\mathbf{I} + (1 - 2y)(\mathbf{W}\mathbf{x})^T) \mathbf{W} \\ \Delta w_0 = \eta(1 - 2y). \end{cases} \quad (8.17)$$

The repelling effect of the anti-Hebbian learning in the multidimensional system (perceptron) has a profound and unexpected consequence: it also repels the outputs from each other because any degenerate weight matrix with  $\det \mathbf{W} = 0$  is an unstable stationary point; that is, the rule naturally seeks

weight vectors that are aligned with the inputs and are orthogonal among each other if the observations are spatially whitened with unit variance (can always be done as a preprocessing step). In the case where the inputs are mixtures of sources, it promotes each output to seek an individual source, and once it locks into it, the other outputs must represent one of the other sources, implementing effectively a joint output maximization criterion, when maximizing simply output entropy. Therefore, in the case where the logistic function is a good estimate of the CDF of the input, maximizing output entropy at the output of the nonlinearity maximizes the mutual information between input and system outputs.

This is a simple and very ingenious learning rule that has many applications, but only under the conditions mentioned performs ICA. We show that ITL can be also used very effectively for ICA however in its general formulation it must estimate the information potential in the joint field.

## 8.7 The Information Bottleneck (IB) Method

Shannon left meaning out of information theory because he was interested in a mathematical description of the concept that suited very well for the design of communication systems. However, many researchers believe that the importance of IT goes beyond this mathematical formalism and can also be useful in characterizing optimal solutions in cases where conditional probabilities are important, as in real-world machine learning and computational neuroscience modeling [171]. The fundamental problem is how to characterize “relevant” information in a signal  $x \in X$  (normally corrupted by noise), which is defined as the information provided with respect to another signal  $y \in Y$ . A very important step towards this goal was recently achieved by Tishby and collaborators with the information bottleneck (IB) method [316]. They formulated a variational principle for the extraction of relevant information by finding the optimal compromise between (1) the minimization of mutual information between  $X$  and its compressed version  $\tilde{X}$ , estimated from rate distortion theory; and (2) the maximization of mutual information between the two variables  $I(\tilde{X}, Y)$ , from channel capacity, where  $Y$  is the relevant variable. The name information bottleneck comes from the bottleneck networks used for PCA, where projection to a subspace and mapping back to the full space finds the eigenspace of the data. In this case it is the lossy compression of  $X$  to  $\tilde{X}$  when used to match  $Y$  in the MI sense that creates the bottleneck and that extracts the relevant information that  $X$  contains with respect to  $Y$ . This formulation builds upon rate distortion theory [17], but takes away one of its fundamental shortcomings because the rate distortion function is built automatically from the optimization of the cost function. Moreover, the method has a solid foundation in statistical mechanics and many important results regarding the method have been recently obtained [305].

## IB Optimization

The fundamental problem in IB is to find a reduced representation  $\tilde{X}$  of  $X$  characterized by the conditional PDF  $p(\tilde{x}|x)$  that preserves the information of  $X$  with respect to another variable  $Y$ . The optimal solution can be found by minimizing the functional

$$\Lambda[p(\tilde{x}|x)] = I(\tilde{X}, X) - \beta I(\tilde{X}, Y), \quad (8.18)$$

where  $\beta$  is the variational parameter, while maintaining  $p(\tilde{x}|x)$  as a true density. If one considers that the foot print of information theory lies between the minimization of MI as in rate distortion theory and its maximization for channel capacity, Eq. (8.18) shows clearly the “information theory” roots of the IB method. Notice that when  $\beta \rightarrow 0$  the quantization approaches the trivial point solution  $\{\tilde{X}\} = 0$ , whereas when  $\beta \rightarrow \infty$  one preserves all the information in  $X$ . It turns out that this variational problem still accepts an exact analytical solution given by [316]

$$\begin{aligned} p(\tilde{x}|x) &= \frac{p(\tilde{x})}{Z(x, \beta)} \exp\left(-\beta \sum_y p(y|x) \log \frac{p(y|x)}{p(y|\tilde{x})}\right) \\ &= \frac{p(\tilde{x})}{Z(x, \beta)} \exp(-\beta D_{KL}(p(y|x)||p(y|\tilde{x}))), \end{aligned} \quad (8.19)$$

where  $Z(x, \beta)$  is the partition function,  $p(\tilde{x}) = \sum_x p(\tilde{x}|x)p(x)$ , and the distribution  $p(y|\tilde{x})$  can be obtained with Bayes’ rule in the Markov chain  $\tilde{X} \leftarrow X \leftarrow Y$  as  $p(y|\tilde{x}) = 1/p(\tilde{x}) \sum_x p(y|x)p(\tilde{x}|x)p(x)$ .

The free parameter in this algorithm is only the variational parameter  $\beta$  that the user has to choose taking into consideration that small  $\beta$  will select small representations (hence little relevance), whereas too large a  $\beta$  will practically use all the structure on  $X$ . For each  $\beta$  there is a convex curve in the  $I_{X\tilde{X}}, I_{Y\tilde{X}}$  plane, analogous to rate distortion curves. Because the distributions are not known explicitly in practice, and the solution has to be consistent among  $p(y|\tilde{x})$ ,  $p(\tilde{x})$ , and  $p(\tilde{x}|x)$ , alternating iterations between them according to

$$\begin{cases} p_n(\tilde{x}|x) = \frac{p_n(\tilde{x})}{Z_n(x, \beta)} \exp(-\beta d(x, \tilde{x})) \\ p_{n+1}(x) = \sum_x p(x)p_n(\tilde{x}|x) \\ p_{n+1}(y|\tilde{x}) = \sum_y p(y|x)p_n(x|\tilde{x}), \end{cases} \quad (8.20)$$

where  $n$  is iteration number and  $d(x, \tilde{x}) = D_{KL}(p(y|x)||p(y|\tilde{x}))$ . The partition function  $Z_n(x, \beta)$  and the densities have to be evaluated at every iteration, which means that the optimization can be time consuming, but it achieves remarkable results when applied to speech [305] and text processing [245].

## 8.8 The Principle of Relevant Information (PRI)

A common framework for the three different aspects of unsupervised learning, namely clustering, principal curves, and vector quantization has been missing until today. Such a framework can only be developed if the central theme of unsupervised learning is addressed, which is to capture the underlying structure in the data. Two schools of thought exist [38]. One approach is to build generative models that effectively describe the observed data. The parameters of these generative models are adjusted to optimize the likelihood of the data with constraints on model architecture. Bayesian inference models and maximum likelihood competitive learning are examples of this line of reasoning [207]. This modeling approach requires some form of regularization to select a proper model order, such as minimum description length (MDL), Bayesian information criterion (BIC), or Akaike's information criterion (AIC). Minimizing the model order forces the network to learn parsimonious representations that capture the underlying regularities (redundancy) in the data.

The second approach uses self-organization principles. As Watanabe clearly stated [330], the goal of learning is to extract information or reduce uncertainty in the data which can be accomplished by minimizing entropy. However, minimization of entropy of a dataset leads, as is well known, to a featureless solution given by the collapse of all the samples to a single point in space. Therefore, without constraints that preserve different levels of detail about the original data it leads to a featureless solution (a single point). As we have seen, the principles of self-organization described by Linsker, Barlow, and Tishby address in different forms information preservation and redundancy reduction. Hence the relevance of information is crucial when applying Watanabe's idea to practical problems, and it gave the name for this new self-organizing principle. The idea here is to construct energy functions that effectively combine these two competing aspects as done in the IB method. Thus by minimizing a combined overall function, we effectively allow the data to self-organize and reveal their structure. Similar attempts have been pursued in the literature, i.e., Heskes [146] developed a free energy functional as a weighted combination of quantization error and entropy terms and explored the exact connection among the self-organizing map (SOM), vector quantization, and mixture modeling.

We choose the second approach to develop a simple yet elegant framework for unsupervised learning with the advantage of not imposing statistical models on the data and instead allowing samples the freedom to interact with each other and through this process reveal the data hidden structure through self-organization. Furthermore, by using ITL algorithms and dealing directly with the PDF, the method effectively goes beyond the second-order statistics and extracts as much information as possible from the data.

## Principle of Relevant Information

The conventional unsupervised learning algorithms (clustering, principal curves, vector quantization) are solutions to an information optimization problem that balances the minimization of data redundancy with the distortion between the original data and the solution, expressed by

$$L[p(x|x_o)] = \min_X (H(X) + \lambda D_{KL}(X||X_o)), \quad (8.21)$$

where  $x_o \in X_o$  is the original dataset, and  $x \in X$  is the compressed version of the original data achieved through processing,  $\lambda$  is a variational parameter, and  $H(X)$  and  $D_{KL}(X||X_o)$  are, respectively, the entropy and the KL divergence between the original and the compressed data.

The principle of relevant information (PRI) shares the spirit of the information bottleneck method, but its formulation addresses the entropy of a single dataset. The solution is specified as an optimization over the conditional of the compressed data given the original data. Each of the classical unsupervised learning algorithms (clustering, principal curves, vector quantization) is specified by a different value of  $\lambda$  in the optimization. One of the appeals of this unifying principle for unsupervised learning is that it is formulated in terms of information-theoretic quantities and fast adaptation ITL algorithms exist as shown below.

There is an interesting cognitive interpretation of the PRI that is worth mentioning. A fundamental problem in memory design is how to store an increasing amount of facts in a system with finite memory resources. The original data  $X_o$  can be considered the system memory of an event, and  $X$  the memory trace through time. Therefore the PRI specifies how the original memory evolves under the effect of two information forces: the need to simplify the memory trace of the stored data (the entropy term) and the need to preserve its similarity to the original self. Different values of  $\lambda$  provide different bits of information of the memory traces with respect to the original memory  $X_o$ . Therefore this can be used as an information-theoretic model for working memory and long-term memory. Many other interpretations are possible.

## Algorithms for PRI

Let us consider the family of compressed data  $x$  obtained by applying Eq. (8.21) to the original data  $x_o$ . The goal of unsupervised learning is to create a hierarchical “skeleton of the dataset”: the modes of the data, the principal curves, and vector-quantized approximations of the data with a specified resolution. All of these special cases can be considered more or less distorted versions of the original dataset. The variational parameter  $\lambda$  in Eq. (8.21) controls the level of distortion in the compressed data, because  $\min D_{KL}(X||X_o)$  measures the allowable distortion of the original (similar to the rate distortion function) under the specified weighting given by  $\lambda$ . The first term pushes

the solution into the data manifold, because minimizing entropy without constraints moves the samples to the center of the data cluster. The major conceptual difference with respect to the IB method is that here we are interested in a different problem – minimizing entropy of a single dataset  $X$  – instead of maximizing mutual information between two datasets, which simplifies the mathematics and the algorithms. Another difference is the use of the KL divergence instead of the mutual information because of internal consistence as discussed below. So PRI is based on entropy and relative entropy instead of MI as the IB method.

An added advantage of this formulation is that we can directly use the estimators of ITL to derive fast fixed-point algorithms to solve for the different solutions. Indeed we can rewrite Eq. (8.21) with Renyi's formulation of entropy and divergence as

$$L[p(x|x_o)] = \min_X (H_\alpha(X) + \lambda D_\alpha(X\|X_o))$$

where Renyi's entropy is defined in Eq. (2.9), and Renyi's divergence or relative entropy in Eq. (2.73). For estimation purposes, we propose to measure redundancy by Renyi's quadratic entropy  $H_2(x)$ , and to measure the distortion between the original and the processed data by the Cauchy – Schwarz PDF divergence  $D_{CS}(X, X_o)$ , because of the equivalence shown in Eq. (2.94). This yields

$$\begin{aligned} J(X) &= \min_X [H_2(X) + \lambda D_{CS}(X, X_o)] \\ &= \min_X [(1 - \lambda)H_2(X) + 2\lambda D_{CEF}(X, X_o) - \lambda H_2(X_o)], \end{aligned} \quad (8.22)$$

where  $D_{CS}(X, X_o) = 2D_{CEF}(X, X_o) - H_2(X) - H_2(X_o)$ , and  $D_{CEF} = -\log V(X, X_o)$  is the logarithm of the cross-information potential used so extensively in Chapter 7 and  $\lambda \in R^+$  is the variational parameter. In Eq. (8.22)  $X$  is the variable that evolves over iterations and hence appears in the argument of the cost function. The last term in Eq. (8.22) is constant with respect to  $X$ , therefore for optimization we can reduce  $J(X)$  to

$$J(X) = \min_X [(1 - \lambda)H_\alpha(X) - 2\lambda \log V(X, X_o)]. \quad (8.23)$$

We further propose to use the information potential (IP) estimator of Renyi's entropy to estimate all these quantities directly from samples. Notice the simplicity of this formulation which is completely described by only two parameters: the weighting parameter  $\lambda$  and the inherently hidden resolution parameter  $\sigma$  in the kernel of the IP. The resolution parameter controls the “scale” of the analysis whereas the weighting parameter effectively combines the regularization and similarity terms in appropriate proportion to capture different aspects of the data structure.

What is interesting is that the traditional unsupervised learning algorithms are obtained for specific values of the weighting parameter as we show next.

Effectively, the PRI produces a self-organizing decomposition of the data in hierarchical features. An added advantage of this formulation for unsupervised learning is that there is a very fast fixed-point algorithm, called the mean shift algorithm as described in Chapter 7, Section 7.6, which can solve this optimization problem efficiently. Let us study in detail the PRI for some landmark values of  $\lambda$ .

### Case 1: A Single Point — Gaussian Blurring Mean Shift

Let us start analyzing Eq. (8.21) for the special case of  $\lambda = 0$ . This would reduce the cost function to

$$J(X) = \min_X H_{R_2}(X) = \min_X (-\log(V(X))). \quad (8.24)$$

If you recall from Section 7.6 this is exactly Eq. (7.28), which we showed is equivalent to the Gaussian blurring mean shift algorithm (GBMS). Thus  $\lambda = 0$  minimizes the overall Renyi's quadratic entropy of the dataset. For kernels with infinite support as the Gaussian that is used here, this leads to a single point as the global solution because then  $H(X) = -\infty$  (for discrete cases  $H(X) = 0$ ). With  $X$  initialized to the original dataset  $X_o$ , successive iterations of this fixed point algorithm would “blur” the dataset ultimately giving a single point.

### Case 2: The Modes — Gaussian Mean Shift

We consider another special case of the cost function Eq. (8.21) when  $\lambda = 1$ . In this scenario the cost function is given by

$$J(X) = \min_X D_{CEF}(X, X_o) = \min_X (-\log(V(X, X_o))) \quad (8.25)$$

which is exactly Eq. (7.31), and as we have demonstrated it corresponds to the Gaussian mean shift algorithm (GMS) that finds the modes of the dataset.

**Theorem 8.1.** *With  $X$  initialized to  $X_o$  in GMS,  $J(X)$  reaches its local maximum at the fixed points of Eq. (8.25).*

*Proof.* Using Eq. (7.33) with a Gaussian kernel the mean shift vector at iteration  $\tau$  is

$$\begin{aligned} x(n+1) - x(n) &= m(x(n)) - x(n) = \frac{\sum_{j=1}^{N_0} G(x(n) - x_o(j))(x_o(j) - x(n))}{\sum_{j=1}^{N_0} G(x(n) - x_o(j))} \\ &= \sigma^2 \frac{\nabla_x \hat{p}_{X_o, \sigma}(x)}{\hat{p}_{X_o, \sigma}(x)} = \sigma^2 \nabla_x \log(\hat{p}_{X_o, \sigma}(x)) \end{aligned} \quad (8.26)$$

We see that in GMS the samples move in the direction of the normalized density gradient with increasing density values. Each sample converges to the mode dictated by the information forces acting on it and that define the convex hull of the mode. Let  $s_l = \{1, \dots, L\}$  be the modes of  $p_{X_o, \sigma}(x)$ . Associate each  $x_i = \{1, \dots, N\}$  with the corresponding mode  $s_{i^*}, i^* \in \{1, \dots, L\}$  to which it converges. Then,

$$\begin{aligned} V(X, X_o) &= \frac{1}{NN_0} \sum_{i=1}^N \sum_{j=1}^{N_0} G_\sigma(x_i - x_{oj}) = \frac{1}{N} \sum_{i=1}^N \hat{p}_{X_o, \sigma}(x_i) \\ &\leq \frac{1}{N} \sum_{i^*=1}^N \hat{p}_{X_o, \sigma}(s_{i^*}) \leq \max_{s_l} \hat{p}_{X_o, \sigma}(s_l). \end{aligned} \quad (8.27)$$

Because the CIP at the fixed points (modes) is maximal,  $J(X) = -\log V(X, X_o)$  reaches its local minimum starting with the initialization  $H(X_o)$ .

### Case 3: Beyond the Modes — Principal Curves and Back to the Original Data

We now consider a more general situation when  $1 < \lambda < \infty$ .

$$\begin{aligned} J(X) &= \min_X [(1 - \lambda)H_\alpha(X) - 2\lambda \log V(X, X_o)] \\ &= \min_X \left[ -(1 - \lambda) \log \left( \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N G_\sigma(x_i - x_j) \right) \right. \\ &\quad \left. - 2\lambda \log \left( \frac{1}{NN_0} \sum_{i=1}^N \sum_{j=1}^{N_0} G_\sigma(x_i - x_{oj}) \right) \right]. \end{aligned} \quad (8.28)$$

Differentiating  $J(X)$  with respect to  $x_k = \{1, \dots, N\}$  and equating it to zero yields

$$\frac{\partial J(X)}{\partial x_k} = \frac{2(1 - \lambda)}{V(X)} F(x_k) + \frac{2\lambda}{V(X, X_o)} F(x; X_o) = 0. \quad (8.29)$$

In this general scenario, the samples of dataset  $X$  move under the influence of two forces: the information force  $F(x_k)$  exerted by all samples of the dataset  $X$  and the “cross” information force  $F(x_k, X_o)$  exerted by the samples belonging to the fixed dataset  $X_o$ . Rearranging Eq. (8.29) gives us the fixed-point update equation as shown in Eq. (8.30)

$$\begin{aligned}
x_k(n+1) &= m(x_k(n)) = c \frac{(1-\lambda)}{\lambda} \frac{\sum_{j=1}^N G_\sigma(x_k - x_j)x_j}{\sum_{j=1}^{N_o} G_\sigma(x_k - x_{oj})} \\
&\quad + \frac{\sum_{j=1}^{N_o} G_\sigma(x_k - x_{oj})x_{oj}}{\sum_{j=1}^{N_o} G_\sigma(x_k - x_{oj})} - c \frac{(1-\lambda)}{\lambda} \frac{\sum_{j=1}^N G_\sigma(x_k - x_j)x_k}{\sum_{j=1}^{N_o} G_\sigma(x_k - x_{oj})}, \\
c &= \frac{V(X, X_o)}{V(X)} \frac{N_o}{N}.
\end{aligned} \tag{8.30}$$

Notice that in this case there are a total of three ways of rearranging  $\partial J(X)/\partial x_k$  to get a fixed-point update rule, but we found only the fixed-point iteration Eq. (8.30) to be stable and consistent.

We have seen earlier that  $\lambda = 1$  gives the modes of the data. As the value of  $\lambda$  is increased more emphasis is given to the similarity between  $X$  and  $X_o$ , that is,  $D_{cs}(X, X_o)$ . Experimentally we found that for two-dimensional datasets the *principal curve* is obtained for  $1 < \lambda < 3$ . Furthermore, for  $\lambda > 3$  in a two-dimensional dataset makes the algorithm go beyond the principal curve and represent more finely all the denser regions of the data, which is the vector quantization regime. In short, every fine nuance of the PDF is captured as the  $\lambda$  value is increased to infinity, yielding back the dataset  $X_o$ . So by varying  $0 < \lambda < \infty$ , PRI goes from a single point to the original dataset. This is neatly summarized in Figure 8.5 for a two-dimensional dataset. Another interesting interpretation comes from careful observation of Eq. (8.22). For values of  $\lambda > 1$ , the first term  $H(X)$  is negative whereas the second term  $H(X, X_o)$  is positive. Thus, the cost function attempts to maximize the entropy and minimizes the  $D_{CS}$  at the same time. These give rise to opposing forces which make the data samples settle in different stationary configurations that correspond to a hierarchical, self-organizing, feature decomposition of the input.

**Theorem 8.2.** *When  $\lambda \rightarrow \infty$ ,  $J(X)$  minimizes the Cauchy–Schwarz divergence  $D_{cs}(X, X_o)$ .*

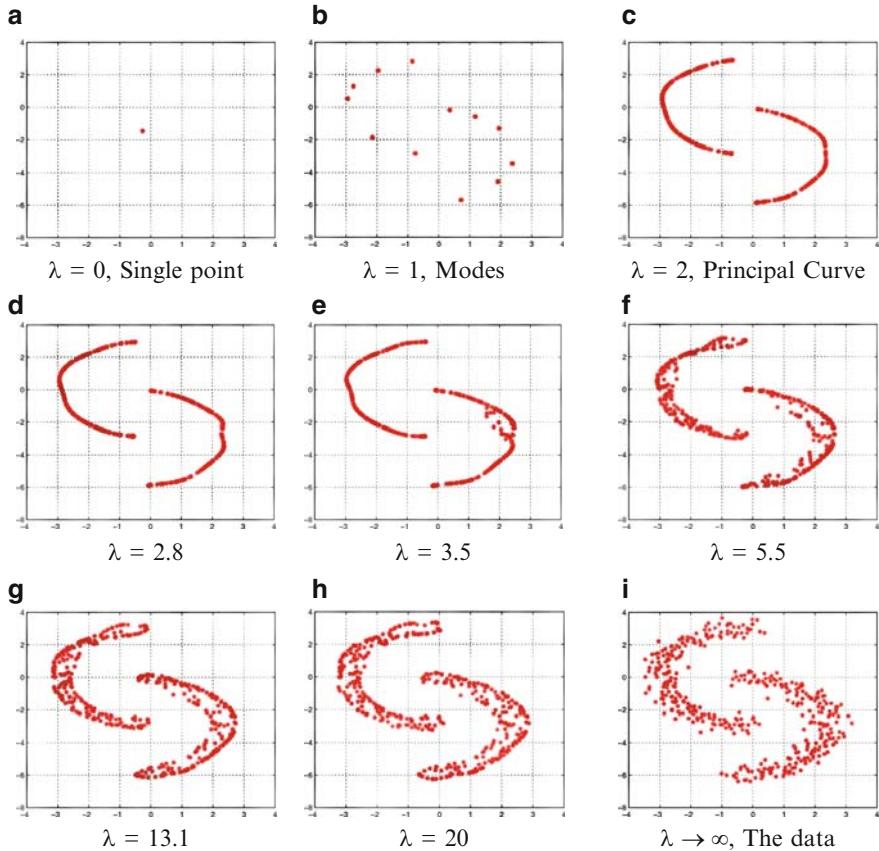
*Proof.* We prove this theorem by showing that the fixed-point equation derived by optimizing  $D_{cs}(X, X_o)$  is the same as Eq. (8.30) when  $\lambda \rightarrow \infty$ . Consider the cost function,

$$F(X) = \min D_{CS}(X, X_o) = \min[-2 \log V(X, X_o) + \log V(X) + \log V(X_o)] \tag{8.31}$$

Differentiating  $F(X)$  with respect to  $x_k = \{1, \dots, N\}$  and equating it to zero would yield the following fixed-point update equation

$$\begin{aligned}
x_k(n+1) &= m(x_k(n)) = -c \frac{\sum_{j=1}^N G_\sigma(x_k - x_j)x_j}{\sum_{j=1}^{N_o} G_\sigma(x_k - x_{oj})} + \frac{\sum_{j=1}^{N_o} G_\sigma(x_k - x_{oj})x_{oj}}{\sum_{j=1}^{N_o} G_\sigma(x_k - x_{oj})} \\
&\quad + c \frac{\sum_{j=1}^N G_\sigma(x_k - x_j)x_k}{\sum_{j=1}^{N_o} G_\sigma(x_k - x_{oj})}, \quad c = \frac{V(X, X_o)}{V(X)} \frac{N_o}{N}
\end{aligned} \tag{8.32}$$

Taking the limit  $\lambda \rightarrow \infty$  in Eq. (8.30) gives Eq. (8.32).



**Fig. 8.5.** An illustration of the structures revealed by the principle of relevant information for the crescent-shaped dataset for  $\sigma^2 = 0.05$ . As the values of  $\lambda$  increase the solution passes through the modes, principal curves and in the extreme case of  $\lambda \rightarrow \infty$  we get back the data themselves as the solution. (from [259]).

**Corollary 8.1.** With  $X$  initialized to  $X_o$ , the data satisfy Eq. (8.32) fixed-point update rule.

*Proof.* When  $X(0) = X_o$ ,  $V(X) = V(X, X_o)$  and  $N = N_o$ . Substituting this in Eq. (8.32) we see that  $x_k(n+1) = x_k(n)$ .

As the  $\lambda$  value becomes very large, the emphasis is laid more and more on the similarity measure  $D_{cs}(X, X_o)$ . Theorem 8.2 and its corollary depict an extreme case where  $\lambda \rightarrow \infty$ . At this stage, PRI gives back the data as the solution.

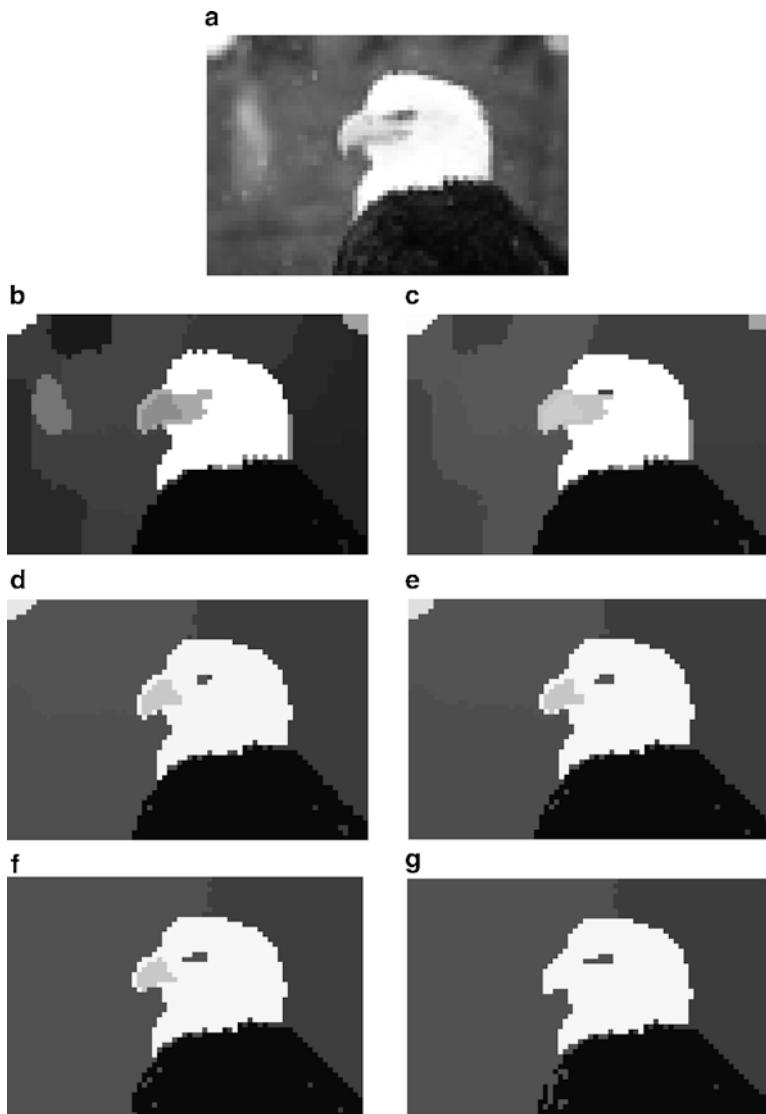
## Image Clustering

Mode-finding algorithms have been used extensively to cluster and segment data into meaningful regions, and we presented them in Chapter 7. However, we just want to recall that both GBMS and GMS are two special cases of PRI with  $\lambda = 0$  and 1, respectively. We illustrate here segmentation of the bald eagle image (Figure 8.6) with  $\lambda = 1$  and compare it with spectral clustering.

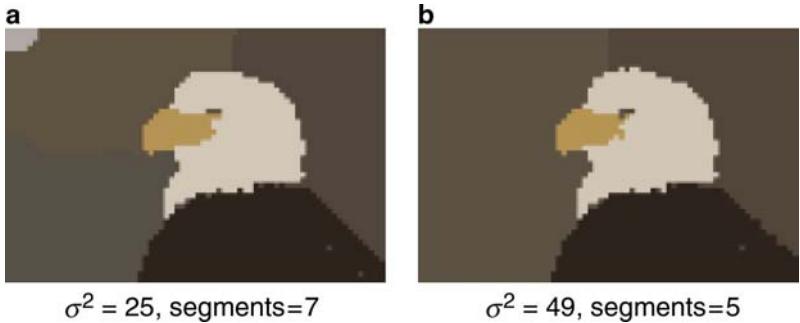
We downsampled this image to  $48 \times 72$  pixels using a bicubic interpolation algorithm, which still preserves the segments very well at the benefit of reducing the computation time for this example because the PRI is  $O(N^2)$  complexity. The incomplete Cholesky decomposition treated in Chapters 2 and 3 is recommended for practical applications of the PRI. The affinity matrix construction in spectral clustering becomes difficult to manage for more than 5000 data points due to memory issues. In order to get meaningful segments in images close to human accuracy, the perceived color differences should correspond to Euclidean distances of pixels in the feature space. One such feature space that best approximates perceptually uniform color spaces is the  $L^*u^*v$  space [337]. Furthermore, the  $x$  and  $y$  coordinates are added to this feature space to take into account the spatial correlation. Thus, the 3456 data points exist in a five-dimensional feature space.

The obtained results for the multiscale analysis are shown in Figure 8.6. Before plotting segments with less than 10 data points were merged to the closest cluster mean. This is needed to eliminate spurious clusters arising due to isolated points or outliers. In the case of the bald eagle image, spurious clusters resulted only for kernel size  $\sigma = 3$ . This is clearly a result of a too narrow kernel size selection. As the kernel size was increased, the number of segments is drastically reduced reaching a level of seven segments for both  $\sigma = 5$  and  $\sigma = 6$ . Note the sharp and clear segments obtained. For example, the eagle itself with its beak, head and torso is very well represented. One can also appreciate the nice hierarchical structure in this analysis where previous disjoint but close clusters merge to form larger clusters in a very systematic way. For  $\sigma = 7$ , five segments are obtained which is the broadest segmentation of this image. Beyond this point, important image segments disappear as shown for  $\sigma = 8$  where the beak of the eagle is lost.

Because both 5 and 7 segments look very appealing with the previous analysis, we show the comparison with spectral clustering for both these segments. Figure 8.7 shows these results for  $\sigma = 5$  and  $\sigma = 7$  respectively. Spectral clustering performs extremely well with clear well defined segments. Two things need special mention. First, the segment boundaries are sharper in spectral clustering when compared to GMS results. This is understandable because in GMS each data point is moved towards its modes and in boundary regions there may be ambiguity as to which peak to climb. This localized phenomenon leads to a pixelization effect at the boundary. Second, it is surprising how the spectral clustering technique could depict the beak region so well in spite of the low-resolution image. A close observation shows that the region of



**Fig. 8.6.** PRI for  $\lambda = 1$ , for different values of kernel size (which yields the segments) (from [259]).



**Fig. 8.7.** Spectral clustering for the bald eagle (from [259]).

intersection between the beak and the face of the eagle has a color gradation. This is clearly depicted in the GMS segmentation result for  $\sigma = 4$  as shown in Figure 8.6c.

### Manifold Learning

The modes of the data are the peaks of the PDF and hence correspond to regions of high concentration of data samples and they are captured by the PRI. We saw that when  $\lambda$  is increased beyond 1, the weight given to the similarity term  $D_{\text{cs}}(X, X_0)$  increases, which is responsible for finding the modes of the data, as shown in Theorem 8.1. Therefore to minimize  $J(X)$  it is essential that modes be part of the solution. On the other hand because  $(1 - \lambda) < 0$  we are effectively maximizing  $H(X)$ , which tends to spread the data. To satisfy this and to minimize  $J(X)$  overall, the compressed data spread along a curve connecting the modes of the PDF. This satisfies our intuition because a curve through the modes of the data has higher entropy (more information) than just discrete points representing modes. One attractive feature of this principle is that the modes, not the mean of the data are anchoring the curve. Notice that modes are points and this curve is one-dimensional, therefore this decomposition yields multidimensional features of the data set. For 2D data this decomposition stops here, but in high-dimensional spaces, when  $\lambda$  is increased beyond 2 (up to the dimensionality of the data minus 1) we are talking about multidimensional nonlinear principal spaces where the data live, are commonly called data manifolds [274].

### A New Definition of Principal Curves

Erdogmus and Ozertem recently introduced a new principal curve definition [96]. We briefly summarize here the definition and the result obtained. Let  $x \in R^N$  be a random vector and  $p(x)$  its PDF. Let  $g(x)$  denote the transpose of the gradient of this PDF and  $U(x)$  its local Hessian.

**Definition 8.1.** A point  $x$  is an element of the  $D$ -dimensional principal set, denoted by  $\rho^D$ , if and only if  $g(x)$  is orthonormal (null inner product) to at least  $(N - D)$  eigenvectors of  $U(x)$  and  $p(x)$  is a strict local maximum in the subspace spanned by these eigenvectors.

Using this definition Erdogmus and Ozertem proved that the modes of the data correspond to  $\rho^0$  (i.e., 0-dimensional principal set) and further more  $\rho^D \subset \rho^{D+1}$ . It is clear that  $\rho^1$  is a one-dimensional curve passing through the data,  $\rho^2$  is a two-dimensional surface, and so on. Thus with  $\rho^D \subset \rho^{D+1}$ , there is a hierarchical structure associated with this definition with modes being part of the curve  $\rho^1$ . This curve can be considered as a non linear principal component that best approximates the data. By going through all the dense regions of the data (the modes),  $\rho^1$  can be considered as a new definition of principal curves that effectively captures the underlying one-dimensional structure of the data.

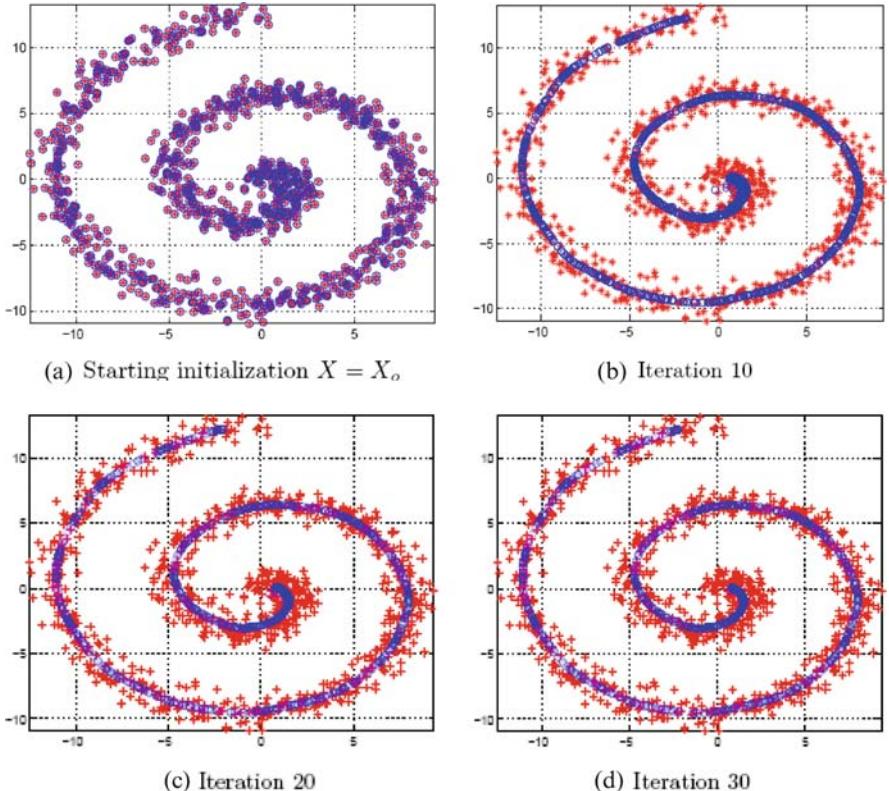
In order to implement this definition, the reconstruction of the PDF using Parzen windows is necessary. Erdogmus first found the modes of the data using the Gaussian mean shift (GMS) algorithm ( $\lambda = 1$ , case of PRI). Starting from each mode and shooting a trajectory in the direction of the eigenvector corresponding to the largest eigenvalue one can effectively trace out this curve. A numerical integration technique such as fourth order Runge–Kutta was utilized to get the next point on the curve starting at the current point and moving in the direction of the eigenvector of the local Hessian evaluated at that point. Principal curves have immense applications in denoising and as a tool for manifold learning. We illustrate with some examples of how this can be achieved with the PRI.

### Spiral Dataset Example

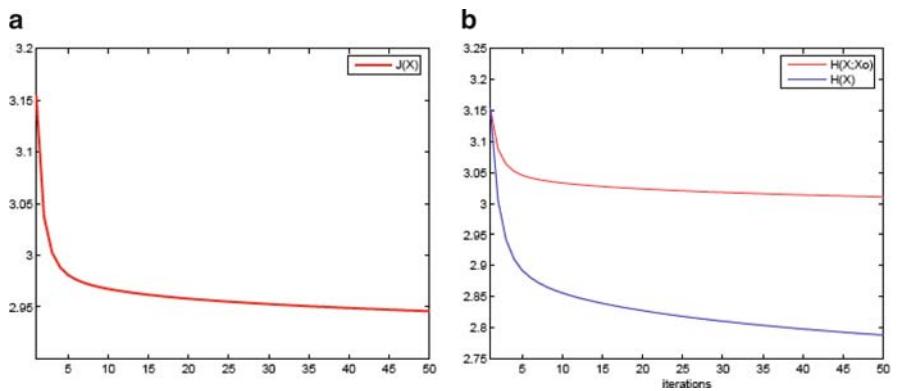
We illustrate the PRI for principal curves with the spiral data in Figure 8.8 which is considered a very difficult problem in principal curves literature [181]. The data consist of 1000 samples perturbed by Gaussian noise with 0.25 variance.

We would like to add here that this is a much noisier dataset compared to the one actually used in [25]. We use  $\lambda = 2$  for our experiment although we found that any value between  $1 < \lambda < 3$  can be actually used. Figure 8.8 shows the different stages of the principal curve as it evolves from  $X = X_0$ .

The samples converge quickly to the curve in the middle of the dataset. By the tenth iteration the structure of the curve is clearly revealed. After this, the changes in the curve are minimal with the samples only moving along the curve (and hence always preserving it). What is even more exciting is that this curve exactly passes through the modes of the data for the same scale  $\sigma$ . Thus our method gives a principal curve that satisfies Erdogmus' definition naturally and does not require PDF estimation. We also depict the development of the cost function  $J(X)$  and its two components  $H(X)$  and  $D_{CS}(X, X_0)$  as the function of the number of iterations in Figure 8.9.



**Fig. 8.8.** The evolution of principal curves starting with  $X$  initialized to the original dataset  $X_0$ . The parameters are  $\lambda = 2$  and  $\sigma^2 = 2$  (from [259]).



**Fig. 8.9.** Changes in the cost function and its two components for the spiral data as a function of the number of iterations: (a) cost function  $J(X)$ ; (b) Two components of  $J(X)$  namely  $H(X)$  and  $D_{CS}(X, X_0)$  (from [259]).

Note the quick decrease in the cost function due to the rapid collapse of the data to the curve. Further decrease is associated with small changes in the movement of samples along the curve.

### Face Image Compression

Here, we present a quantization result with 64 centers of the edges taken from a face. Apart from being an integral part of image compression, vector quantization also finds application in face modeling and recognition. We would like to preserve the facial details as much as possible, especially the eyes and ears which are more complex features. Previous work on information-theoretic vector quantization (ITQV) by Lehn-Schioler et al. [197] proposed a gradient method to implement ITVQ. The cost function was exactly the  $D_{CS}$  and the gradient update is given by

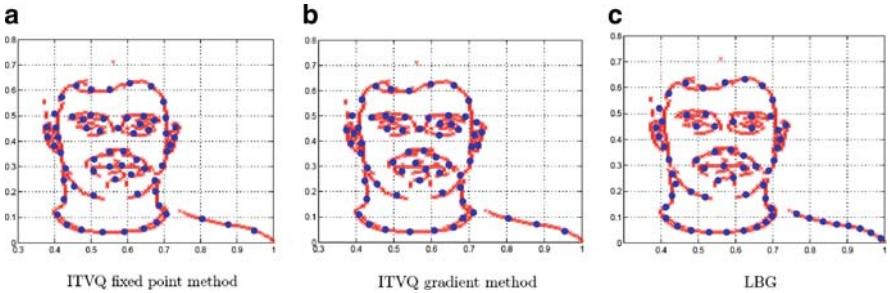
$$x_k(n+1) = x_k(n) - \eta \frac{\partial D_{CS}(X, X_0)}{\partial x_k}, \quad (8.33)$$

where  $\eta$  is the learning rate. Thus the derivative of  $J(X)$  would be equal to

$$\frac{\partial D_{CS}(X, X_0)}{\partial x_k} = \frac{2}{V(X; X_0)} F(x_k; X_0) - \frac{2}{V(X)} F(x_k), \quad (8.34)$$

which corresponds to the PRI of Eq. (8.29) for  $\lambda \rightarrow \infty$ . Unfortunately, as is true in any gradient method, the algorithm almost always gets stuck in local minima, so annealing of the parameters ( $p_1$  = stepsize and  $p_2$  = kernel size) is necessary according to the rule  $p_{in} = k_i p_{i0} / (1 + \gamma_i k_i n)$ , where  $n$  is the iteration number. We repeat the procedure of finding the best parameters for the gradient method. After an exhaustive search we end up with the following parameters:  $k_1 = 1$ ,  $\gamma_1 = 0.1$ ,  $p_{10} = 1$ ,  $k_2 = 1$ ,  $\gamma_2 = 0.15$ , and  $p_{20}$  was set to a diagonal matrix with variance along each feature component equal to the diagonal entries. Although the cost functions coincide, the advantage of the PRI is that we now have a fixed-point solution without a stepsize.

Both algorithms were initialized with 64 random points inside a small square grid centered on the mouth region of Figure 8.10. As done before for the fixed point, the kernel parameters were set exactly as the gradient method for a fair comparison. Notice how the code vectors spread beyond the face region initially with large kernel size and immediately capture the broad aspects of the data. As the kernel size is decreased the samples then model the fine details and give a very good solution. Figure 8.10 shows the results obtained using the PRI, ITVQ gradient method, and the Linde–Buzo–Gray (LBG) vector quantization algorithm [110]. Some important observations can be made. Firstly, among the ITVQ algorithms the fixed point represents the facial features better than the gradient method. For example, the ears are very well modeled in the fixed-point method. Secondly, and perhaps the most important advantage of the fixed-point method over the gradient method is



**Fig. 8.10.** Comparison of different vector quantizers in a human face (from [259]).

**Table 8.1.** Comparison of Clustering Methods (From [259])

Method	$J(X)$	MSE
PRI	0.0253	$3.355 \times 10^{-4}$
ITVQ-grad	0.0291	$3.492 \times 10^{-4}$
LBG	0.1834	$3.05 \times 10^{-4}$

the speed of convergence. We found that the PRI solution was more than five times faster than its gradient counterpart. In image applications, where the number of data points  $N$  is generally large, this translates into a huge saving of computational time.

Thirdly, both ITVQ and PRI algorithms outperform LBG in terms of facial feature representation. The LBG uses many code vectors to model the shoulder region and few code vectors to model the eyes or ears. This may be due to the fact that LBG just uses second-order statistics whereas ITVQ uses all higher-order statistics expressed by the kernel to better extract the information from the data and allocate the code vectors to suit their structural properties. This also explains why ITVQ methods also perform very close to LBG from the MSE point of view as shown in Table 8.1.

On the other hand, notice the poor performance of LBG in terms of minimizing the cost function  $J(X)$ . Obviously, the LBG and ITVQ fixed-point give the lowest values for their respective cost functions.

### PRI Space for Image Processing

Looking closely at the principle of relevant information we see that its two free parameters define a two-dimensional space: the scale parameter given by the kernel size and the variational parameter, where both are positive quantities. This continuous parameter space can be thought of as a control space for unsupervised learning, in the sense that all the points of the space correspond to a pair of values that dictate a transformation of the input image. Unsupervised learning algorithms live in horizontal lines of constant  $\lambda$ . In fact, when we do clustering  $\lambda = 1$  and the kernel size is changed to give different “looks” of the

data at different spatial resolutions. With the PRI we are not limited anymore to horizontal cuts, because we have a cost function that depends on the two parameters, so it is possible for instance to anneal both parameters during processing. Perhaps even more interesting, these parameters can be adapted on the fly by the adaptive system for goal-driven image understanding.

## 8.9 Self-Organizing Principles with ITL Estimators

### ITL for MaxEnt

The manipulation of entropy subject to constraints as developed in MaxEnt and MinXent, is not the most appropriate for learning problems that include training the system parameters. In such cases, an explicit cost function that can be estimated from data and gradient descent learning or similar procedures is preferable. Here we start by demonstrating how we can maximize output entropy to train system parameters with local learning algorithms. The closest second-order statistical analogy is the maximization of the output power, which is the essence of principal component projections, or in more general terms, correlation learning also called Hebbian learning [253].

Hebbian learning is also one of the mainstays of biologically inspired neural processing. Hebb's rule is biologically plausible, and it has been extensively utilized in both computational neuroscience and in unsupervised training of neural systems [54]. In these fields, Hebbian learning became synonymous for correlation learning. But it is known that correlation is a second-order statistic of the data, so it is sub-optimal when the goal is to extract as much structure as possible from the sensory data stream when the data are not Gaussian distributed. Ironically, Donald Hebb never spoke of correlation when he enunciated his famous learning principle, as follows [144]. "When an axon of cell A is near enough to excite cell B or repeatedly or consistently takes part in firing it, some growth or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased." This principle has been "translated" in the neural network and computational neuroscience literature according to the conventional wisdom of second-order statistics as: in Hebbian learning, the weight connecting a neuron to another is incremented proportionally to the product of the input to the neuron and its output [141]. This formulation then can be shown to maximize the correlation between the input and the output of the neuron whose weights are updated through the use of stochastic gradient on a correlation-based cost function [54]. For this reason, correlation has been declared the basis of biological learning, and Hebb's rule has mostly shaped our understanding of the operation of the brain and the process of learning. However, we have to point out that there is a huge assumption of translating Hebb's postulate as the statistical concept of correlation (other researchers have modeled temporal dependent plasticity as InfoMax [52]). We show below that Hebb's postulate can be translated in the

form of a maximum entropy principle with practical advantages. Instead of using the solution proposed by Jayne's using Lagrange multipliers, we directly maximize entropy at the output of a linear or nonlinear system using ITL principles.

### Relation Between the SIG Algorithm and Hebbian Learning

It is straightforward to derive the incremental weight change for the Adaline trained with output maximum entropy, and this was already implicitly done in Chapter 4 when adapting FIR filters. The Adaline is an adaptive linear element followed by a nonlinearity, but the error is calculated before the nonlinearity. We refer to the material of Section 4.3, where the gradient descent update on Shannon's entropy error cost function was presented (Eq. (4.19)). If the goal is to maximize entropy at the output of a linear system, there is no desired signal per se, only the output of the system, so the IP of interest is built from the output samples  $\{y_i\}_{i=1,\dots,N}$ , yielding  $V_\alpha(y)$  instead of the IP of the error sequence. Furthermore, we showed in Chapter 4 (Theorem 4.1) that the expected value of the SIG(L) is an unbiased estimator of Shannon entropy. In an online adaptation scenario, we approximate SIG(L) by its current sample estimate given by SIG(1). With this modification and for Gaussian kernels, the SIG(1) gradient update to maximize the output entropy of an Adaline becomes

$$\frac{\partial \hat{H}_\alpha(y(k))}{\partial w} = \frac{1}{\sigma^2}(y(k) - y(k-1)) \cdot (x(k) - x(k-1)). \quad (8.35)$$

We clearly see that the basic form of the Hebbian update is present (product of inputs and outputs), but now it is applied to the instantaneous increments of the output and the input of the Adaline. With this rule, the neuron implements Hebb's information learning rather than merely correlation learning. Alternatively, we can say that it is possible to approximate the maximization of output entropy by maximizing the correlation between input and output samples of the corresponding time index and minimizing the correlation between noncorresponding time indices.

Now, consider the general case where any differentiable symmetric kernel function may be used in Parzen windowing. For general even symmetric, unimodal, and differentiable kernels that are PDFs themselves, we get the following update rule

$$\frac{\partial \hat{H}_\alpha(y(k))}{\partial w} = f(y(k) - y(k-1)) \cdot (x(k) - x(k-1)), \quad (8.36)$$

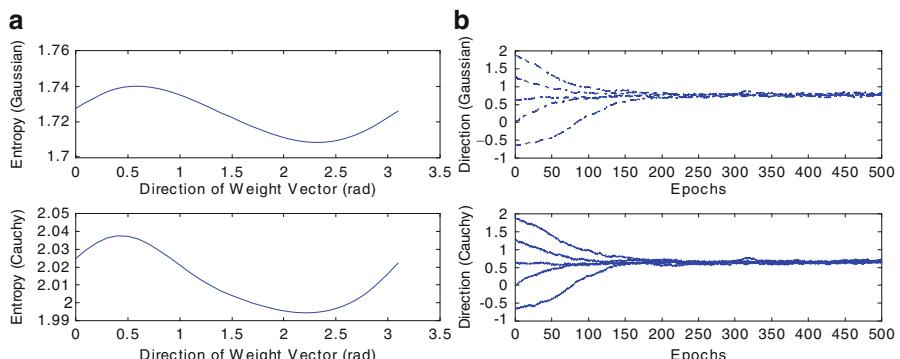
where  $f(x) = -\kappa'_\sigma(x)/\kappa_\sigma(x)$ , where  $\kappa(\cdot)$  is the Parzen kernel and satisfies the condition  $\text{sign}(f(x)) = \text{sign}(x)$ . Thus, the update amount that would be applied to the weight vector is still in the same direction as would be in Eq. (8.35); however, it is scaled nonlinearly depending on the value of the

increment that occurred in the output of the neuron. This is in fact consistent with Hebb's postulate of learning, and demonstrates that the weight adaptation based on the product of the output and the input signals as done today is not the only possibility to implement Hebb's postulate of learning.

### Equal Sample Covariance Spaces

This example demonstrates that the just-derived Hebb's MaxEnt rule extracts more structure from data than the traditional Hebb's rule. Fifty samples of a two-dimensional random vector are generated such that the  $x$ -coordinate is uniform, the  $y$ -coordinate is Gaussian, and the (sample) covariance matrix is identity in the  $x, y$  space. In this case, PCA direction is ill defined, and an online algorithm will find any direction dependent upon the initial conditions, because the variance along each direction is the same. On the other hand, using Hebb's MaxEnt rule, the system can extract the direction along which the data exhibit the most uncertainty (or the least if minimizing entropy), which is the  $y$ -coordinate (the direction of the Gaussian). Figure 8.11a shows the estimated entropy versus direction of the weight vector of a two-input – one-output linear network trained with Eq. (8.35) for Gaussian and Cauchy kernels. Figure 8.11b shows the convergence of weights from different initial conditions to 0.5 radians ( $\pi/2$ ), because the Gaussian distribution has the largest entropy among distributions of fixed variance [16].

There is a main conclusion from this entropy study that we would like to address. In fact, Hebbian MaxEnt learning adapts the weights (synapses) with information forces, and therefore the adaptive system (neural assemblies) is manipulating entropy, which from the point of information processing estimates the uncertainty in the data. As with any mathematical theory, training a linear network with an algorithm that estimates the entropy gradient, predictions can be made. In order to study the realism of Eq. (8.36) to model



**Fig. 8.11.** (a) Entropy vs. direction for the Gaussian and Cauchy kernels, respectively, on a trained system; (b) convergence from different initial conditions for both kernels. (from [89]).

neuronal interactions in the brain, experiments can be set up to see if in fact synaptic strength is only dependent upon the level of excitability of the neuron or if, as the formula predicts, the efficacy of the synapse is also modulated by past action potentials. If this prediction is correct, then we have a very gratifying result that increases even further our admiration for biological processes.

### MaxEnt for Nonlinear Networks and Arbitrary Distributions

Another useful maximum entropy cost characteristic is that it can be easily integrated with nonlinear networks as was demonstrated in Chapters 5 and 6 (MEE-BP). This opens the door for many applications in nonlinear signal processing.

### From PCA to Principal Curves

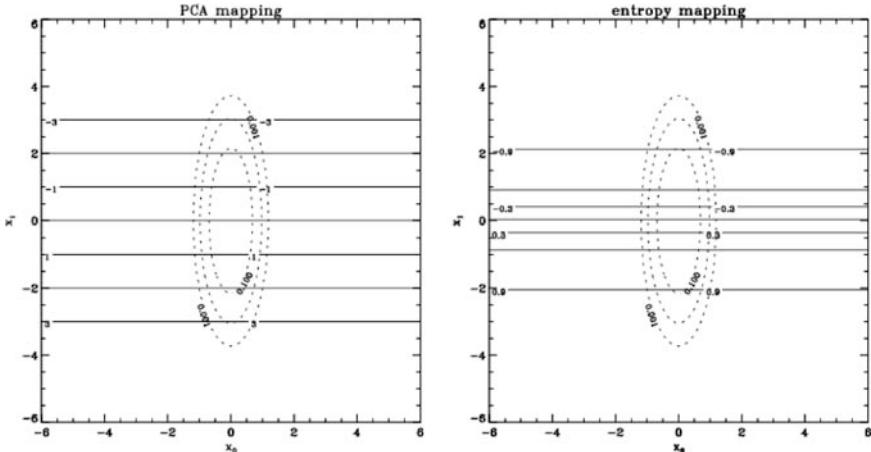
As is well known, the direction of the Adaline weights describes the direction of a hyperplane that in the case of correlation learning defines the first principal component of the input data. The Adaline weights trained with entropy define the direction that maximizes entropy, which in general is different (except if the data are Gaussian distributed).

What happens now if instead of a linear system, we use a nonlinear system such as a MLP and train it to maximize entropy? The flexibility of the new architecture is able to find curves, not necessarily straight lines that maximize entropy. In particular if the data are multimodal, this may much better represent the manifold that contains the data. We begin with the simple case of a two-dimensional Gaussian distribution with zero mean and a covariance matrix given by:

$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 0.1 \end{bmatrix}.$$

The contour plot of this distribution is shown in Figure 8.12 along with the image of the first principal component vector superimposed (left panel). We see from the figure that the first principal component lies along the  $x$ -axis. We draw a set of observations (50 in this case) from this distribution and compute a mapping using an MLP with the entropy maximizing criterion described in the previous section. The architecture of the MLP is 2, 4, 1 indicating 2 input nodes, 4 hidden nodes, and 1 output node. The nonlinearity used is the hyperbolic tangent function. Backpropagation of information forces was utilized (see Section 5.2) with minimization of  $V(y)$ . The learning and annealing rates and the kernel size ( $\sigma = 0.5$ ) were experimentally determined.

The network has the capability to nonlinearly transform the two-dimensional input space onto a one-dimensional output space. The plot at the right of Figure 8.12 shows the image of the maximum entropy mapping onto the input space. From the contours of this mapping we see that the



**Fig. 8.12.** Variance versus entropy, Gaussian case: left image PCA features shown as contours of constant power; right, entropy mapping shown as contours of constant entropy. (from [253]).

maximum entropy mapping lies essentially in the same direction as the first principal components. This result is expected, inasmuch as it illustrates that when the Gaussian assumption is supported by the data, maximum entropy and PCA are equivalent. This result has been reported by many researchers.

We conduct another experiment to illustrate the difference when observations are drawn from a random source whose underlying distribution is not Gaussian. Specifically the PDF is a mixture of Gaussian modes  $f(x) = \frac{1}{2}(G(\mathbf{m}_1, \Sigma_1) + G(\mathbf{m}_2, \Sigma_2))$  where  $G(\mathbf{m}, \Sigma)$  are Gaussian distributions with mean and covariance

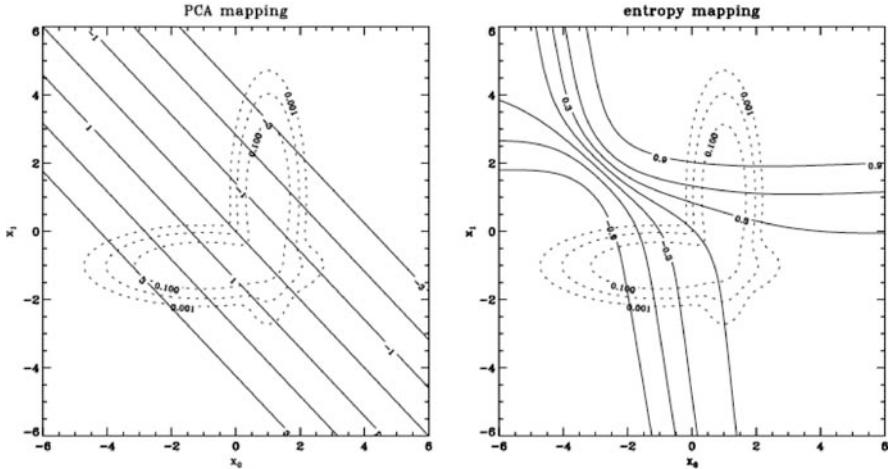
$$\begin{aligned}\mathbf{m}_1 &= \begin{bmatrix} -1.0 \\ -1.0 \end{bmatrix} & \Sigma_1 &= \begin{bmatrix} 1 & 0 \\ 0 & 0.1 \end{bmatrix} \\ \mathbf{m}_2 &= \begin{bmatrix} 1.0 \\ 1.0 \end{bmatrix} & \Sigma_2 &= \begin{bmatrix} 0.1 & 0 \\ 0 & 1 \end{bmatrix}.\end{aligned}$$

In this specific case it can be shown that the principal components of this distribution are the eigenvectors of the matrix

$$\mathbf{R} = \frac{1}{2} (\Sigma_1 + \mathbf{m}_1 \mathbf{m}_1^T + \Sigma_2 + \mathbf{m}_2 \mathbf{m}_2^T) = \begin{bmatrix} 0.62 & 1 \\ 1 & 0.62 \end{bmatrix}.$$

We draw the same 50 samples from these distributions and train the same MLP (2,4,1) with backpropagation to maximize entropy. The contour plot (curves of equal variance) computed with PCA are shown in the left panel of Figure 8.13.

The right panel shows the contours in the input space of constant output entropy for the network trained to maximize entropy. As can be observed, the

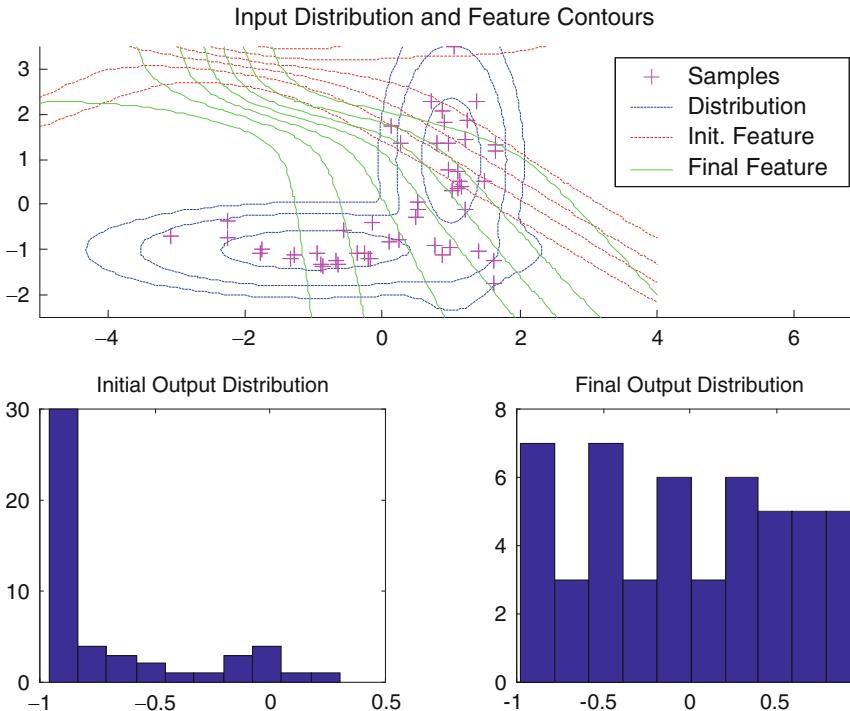


**Fig. 8.13.** Variance versus entropy, non-Gaussian case: left, image of PCA shown as contours of equal power; right, entropy mapping shown as contours of equal entropy ( $\sigma = 0.5$ ) (from [253]).

maximum entropy mapping is more tuned to the structure of the data, but training the network is rather difficult because there is a strong local minimum corresponding to the linear solution. Notice also that the contours of constant entropy over the data basically follow the orthogonal directions of maximum sample density, which is compatible with the concept of principal curves [96]. This experiment helps to illustrate the differences between PCA and entropy mappings. PCA is primarily concerned with finding the covariance eigen directions and is only sensitive to second-order statistics of the underlying data, whereas entropy better explores the structure of the data class. To verify how close the solution provided by the MLP trained with SIG is to the optimal solution (flat distribution because the range of  $y$  is finite), Figure 8.14 shows the output histogram obtained with a representative initial condition for the network. Although the output distribution is not perfectly flat, it is a reasonable approximation given the small size of the dataset.

### Blind Deconvolution in Communication Channels

Another practical application of ITL in self-organization is the minimization of output entropy in communication channels. At first this may seem a paradox to train a system to minimize entropy, inasmuch as it drives the system output towards a single value (for discrete random variables). However, when the system is nonautonomous (i.e., when the input is permanently applied such as a communication receiver), this criterion is useful to counteract the Gaussianization effects of the channel on the transmitted message. Suppose we have

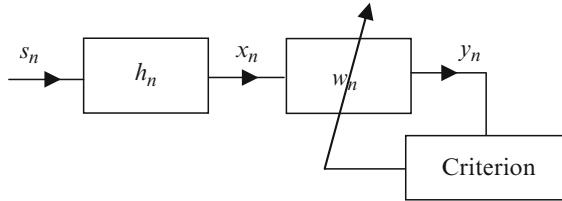


**Fig. 8.14.** Initial and final output distribution and corresponding feature directions in the input space (from [252]).

the result of the convolution of two discrete-time sequences  $s_n$  and  $h_n$ , and let this new sequence be denoted by  $x_n$  (i.e.,  $x_n = h_n^* s_n$ ). In a realistic context, these two sequences could be the input signal to a linear time-invariant (LTI) channel and the channel impulse response. Assuming the channel is linear and has memory, the convolution will skew the input density towards the Gaussian (by the law of large numbers), which happens to be the distribution that has the largest entropy for a fixed variance. Therefore, minimizing the entropy of the output will “undo” the effect of the channel in the messages, as pointed out in [77]. Let us then see how this procedure can be implemented with ITL algorithms.

Suppose that we have the output measurements  $x_n$ , but know neither the channel’s impulse response nor the input sequence. In the classical literature, the problem of determining the input sequence using only this given information (up to an uncertainty in the sign, amplitude, and the delay) is referred to as *blind deconvolution (BD)*.

Typically, BD is represented by the block diagram given in Figure 8.15. Usually the equalizer  $w_n$  is parameterized as an FIR filter and a suitable criterion is determined depending on assumptions of the input signal statistics.



**Fig. 8.15.** Typical blind deconvolution/equalization scheme ( $h_n$  and  $s_n$  are unknown).

Although in reality the measurement samples  $x_n$  may be corrupted by additive noise, this is not considered in our simulations.

There are mainly two approaches that make use of entropy as the adaptation criterion: (1) minimum entropy deconvolution (mED) and (2) maximum entropy deconvolution (MED). MED requires that an estimate of the input signal density be known, from which a nonlinearity created with its cumulative density function can be created, and Bell and Sejnowski method (or equivalent) implemented. In mED these assumptions are not necessary, so it is “truly blind.” Both approaches assume that the samples  $s_n$  are i.i.d, and the equalizer structure is chosen such that it can successfully (even if approximately) invert the channel filter. Here we only address mED and apply it at the output of the receiver. Due to the Benveniste–Goursat–Ruget theorem [33], the entropy of the output of the equalizer is minimized if and only if the overall filter,  $h_n^* w_n$ , is an impulse (with arbitrary delay and scale). The main intuition is that, as the overall filter departs from an impulse, the output distribution approaches a Gaussian density due to the addition of random variables in the filter delay line under the assumption that the input is well modeled by a white random process. It is known that under fixed variance, the Gaussian density has the maximum entropy and thus minimizing entropy under a fixed variance constraint maximizes non-Gaussianity.

In mED the scale factor is an indeterminacy, and in fact there are infinitely many solutions for the optimal weight vector. Normally one imposes a constant variance constraint, that is, by constraining the weight vector to be unit norm at all times. There are two ways to achieve this: normalize the weight vector after each weight update (as Oja’s rule for PCA, e.g.) [233], which is not pursued here because it is more complex in an adaptation scenario. A second approach is to use a scale invariant cost function, so that the performance evaluations of two weight vectors that are colinear but different in norm and sign yield the same value, thus none will be preferred over the other. The entropic cost functions developed in Chapters 3 and 4 do not obey this condition, so they need to be modified.

### Minimum Entropy Deconvolution Using Renyi's Entropy

As an initial step, consider the following theorem, which gives the relationship between the entropies of linearly combined random variables.

**Theorem 8.3.** *Let  $S_1$  and  $S_2$  be two independent random variables with PDFs  $p_{S_1}(\cdot)$  and  $p_{S_2}(\cdot)$ , respectively. Let  $H_\alpha(\cdot)$  denote the order- $\alpha$  Renyi's entropy for a continuous random variable. If  $a_1$  and  $a_2$  are two real coefficients  $Y = a_1 S_1 + a_2 S_2$ , then*

$$H_\alpha(Y) \geq H_\alpha(S_i) + \log |a_i|, \quad i = 1, 2. \quad (8.37)$$

*Proof.* Because  $S_1$  and  $S_2$  are independent, the PDF of  $Y$  is given by

$$p_Y(y) = \frac{1}{|a_1|} p_{S_1}(y/a_1) * \frac{1}{|a_2|} p_{S_2}(y/a_2). \quad (8.38)$$

Recall the definition of Renyi's entropy for  $Y$  and consider the following identity.

$$\begin{aligned} e^{(1-\alpha)H_\alpha(Y)} &= \int_{-\infty}^{\infty} p_Y^\alpha(y) dy \\ &= \int_{-\infty}^{\infty} \left[ \int_{-\infty}^{\infty} \frac{1}{|a_1 a_2|} p_{S_1}\left(\frac{\tau}{a_1}\right) p_{S_2}\left(\frac{y-\tau}{a_2}\right) d\tau \right] dy \end{aligned} \quad (8.39)$$

Using Jensen's inequality for convex and concave cases, we get

$$\begin{aligned} e^{(1-\alpha)H_\alpha(Y)} &\stackrel{\alpha \geq 1}{\leq} \int_{-\infty}^{\infty} \left[ \int_{-\infty}^{\infty} \frac{1}{|a_1|} p_{S_1}\left(\frac{\tau}{a_1}\right) \left[ \frac{1}{|a_2|} p_{S_2}\left(\frac{y-\tau}{a_2}\right) \right]^\alpha d\tau \right] dy \\ &\stackrel{\alpha < 1}{\geq} \int_{-\infty}^{\infty} \frac{1}{|a_1|} p_{S_1}\left(\frac{\tau}{a_1}\right) \left[ \int_{-\infty}^{\infty} \left[ \frac{1}{|a_2|} p_{S_2}\left(\frac{y-\tau}{a_2}\right) \right]^\alpha dy \right] d\tau \\ &= \int_{-\infty}^{\infty} \frac{1}{|a_1|} p_{S_1}\left(\frac{\tau}{a_1}\right) V_\alpha(a_2 S_2) d\tau \\ &= V_\alpha(a_2 S_2) \cdot \int_{-\infty}^{\infty} \frac{1}{|a_1|} p_{S_1}\left(\frac{\tau}{a_1}\right) d\tau = V_\alpha(a_2 S_2) \end{aligned} \quad (8.40)$$

Rearranging terms in the last inequality and using the relationship between entropy and information potential, regardless of the value of  $\alpha$  and the direction of the inequality, we arrive at the conclusion  $H_\alpha(Y) \geq H_\alpha(S_i) + \log |a_i|$ ,  $i = 1, 2$ . An immediate extension of this theorem is obtained by increasing the number of random variables in the linear combination to  $n$ .

**Corollary 8.2.** If  $Y = a_1S_1 + \dots + a_nS_n$ , with i.i.d.  $S_i \sim ps(\cdot)$ , then the following inequality holds for the entropies of  $S$  and  $Y$ .

$$H_\alpha(Y) \geq H_\alpha(S) + \frac{1}{n} \log |a_1 \dots a_n|, \quad (8.41)$$

where equality occurs iff  $a_i = \delta_{ij}$ , where  $\delta$  denotes the Kronecker-delta function.

*Proof.* It is trivial to generalize the result in Theorem 8.3 to  $n$  random variables using mathematical induction. Thus, for the case where all  $n$  random variables are identically distributed we get  $n$  inequalities.

$$\begin{aligned} H_\alpha(Y) &\geq H_\alpha(S) + \log |a_1| \\ H_\alpha(Y) &\geq H_\alpha(S) + \log |a_2| \\ &\vdots \\ H_\alpha(Y) &\geq H_\alpha(S) + \log |a_n| \end{aligned} \quad (8.42)$$

Adding these inequalities, we get the desired result. The necessary and sufficient condition for the equality of entropies is obvious from the formulation. If  $a_i = \delta_{ij}$ , then  $Y = S$ ; therefore the entropies are equal. If  $a_i \neq \delta_{ij}$ , then due to Theorem 8.3, entropy of  $Y$  is greater than the entropy of  $S$  (assuming normalized coefficients). Notice that this corollary does not necessarily guarantee that the overall entropy of  $Y$  is greater than the entropy of  $S$ , because when the absolute value of the product of the gains is less than one, the logarithm brings in a negative component. Through complicated algebra, which we omit here, we are able to show that in the vicinity of a combination where only one of the gains is close to one and all the others are close to zero, these terms lose their significance and the inequality is mostly dominated by the two entropy terms.

Notice that the blind deconvolution problem is structurally very similar to the situation presented in the above corollary. The coefficients  $a_i$  of the linear combination are replaced by the impulse response coefficients of the overall filter  $h_n^*w_n$ . In addition, the random variables  $S$  and  $Y$  are replaced by the source signal and the deconvolving filter (equalizer) output, respectively. Especially, when close to the ideal solution, that is, when  $h_n^*w_n$  is close to an impulse, the second term in Corollary 8.1 will approach rapidly to zero and the two entropy values will converge as the two signals  $Y$  and  $S$  converge to each other. With this understanding, we are ready to go back to the blind deconvolution cost function.

### Renyi's Entropy Implementation

The entropy of a random variable is not scale invariant. In order to solve the blind deconvolution problem using unconstrained optimization techniques and

without having to normalize the weights at every iteration, the original EEC criterion of Chapter 3 must be modified by introducing appropriate terms to make it scale-invariant. For this purpose, consider the following modified cost function.

**Property 8.1.** The modified EEC cost function is scaled-invariant

$$J(Y) = H_\alpha(Y) - \frac{1}{2} \log [\text{Var}(Y)] \quad (8.43)$$

that is,  $J(aY) = J(Y)$ ,  $\forall a \in R$ .

*Proof.* It is trivial to show by a simple change of variables in the integral that for Renyi's entropy (as for Shannon's entropy), we have the following identity between the entropies of two scaled random variables,

$$H_\alpha(aY) = H_\alpha(Y) + \log |a|, \quad (8.44)$$

where we can replace  $\log |a|$  with  $1/2 \log a^2$ . We also know that for variance

$$\text{Var}(aY) = a^2 \text{Var}(Y). \quad (8.45)$$

Combining these two identities, the terms with  $a$  cancel out and we get the desired result. In practice, the information potential replaces the actual entropy expression in the cost function given in Eq. (8.43). The sample variance estimators already satisfy the scaling property of the variance given in Eq. (8.45), however, we saw in Chapter 2 that in order for the entropy estimator to satisfy the scaling property of entropy given in Eq. (8.44), the kernel size needs to be scaled up by the same ratio as the scaling of the norm of the weight vector. Therefore, a kernel size that yields a unit norm weight vector is selected and the weight vector initialized to unit norm. During the course of adaptation, the kernel size is scale up/down according to the new norm of the weight vector.

Practically, because the deconvolving filter  $w_n$  is a causal FIR filter, after the addition of a sufficiently long delay line (length  $L$ ) due to the nonminimum phase situation, one can express its output as a linear combination of the input samples at consecutive time steps as  $y_k = \mathbf{w}^T \mathbf{x}_k$  where the weight vector  $w = [w_0, \dots, w_{2M}]^T$  consists of the FIR impulse response coefficients and  $X_k = [x_k, \dots, x_{k-2M}]^T$  consists of the most recent values of the input signal to the filter. As for the variance term in Eq. (8.44), under the assumption that the source signal is zero-mean and stationary, the unknown channel is linear time-invariant, we can write

$$\text{Var}(Y) = \text{Var}(X) \cdot \sum_{i=0}^{2M} w_i^2 \quad (8.46)$$

Substituting Eqs. (2.18) and (8.46) in Eq. (8.43), we get the nonparametric estimate of the cost function as

$$\hat{J}(w) = \frac{1}{1-\alpha} \log \frac{1}{N^\alpha} \sum_{j=1}^N \left( \sum_{i=1}^N \kappa_\sigma(y_j - y_i) \right)^{\alpha-1} - \frac{1}{2} \log \sum_{i=0}^M w_i^2. \quad (8.47)$$

where  $\text{Var}(X)$  dropped out because it does not depend on the weights of the adaptive filter. Now, the gradient of the EEC scale invariant cost function in Eq. (8.47) with respect to the weight vector is obtained as

$$\frac{\partial \hat{J}}{\partial \mathbf{w}} = -\frac{\sum_{j=1}^N \left( \sum_{i=1}^N \kappa_\sigma(y_j - y_i) \right)^{\alpha-2} \left( \sum_{i=1}^N \kappa'_\sigma(y_j - y_i) (\mathbf{x}_j^T - \mathbf{x}_i^T) \right)}{\sum_{j=1}^N \left( \sum_{i=1}^N \kappa_\sigma(y_j - y_i) \right)^{\alpha-1}} - \frac{\mathbf{w}^T}{\mathbf{w}^T \mathbf{w}}, \quad (8.48)$$

where  $\kappa'_\sigma(\cdot)$  is the derivative of the kernel function with respect to its argument. Given  $N$  samples of  $X_k$ , the adaptive filter may be trained to converge to the inverse of the channel. Choosing a sufficiently small window length of  $N$  samples (depending on the computational requirements), which may be sliding or nonoverlapping, it is possible to estimate the source signal online. As for the optimization techniques that can be applied to obtain the optimal solution, simple gradient descent, conjugate-gradient, Levenberg–Marquardt, or other approaches may be taken as explained in Chapter 5. If the kernel size is chosen sufficiently large (usually, a kernel width that covers about ten samples on the average yields good results), then the performance surface is reasonably simple to search as demonstrated with numerous simulations [93].

### Blind Deconvolution Case Study

In order to test the performance of the proposed blind deconvolution algorithm, we performed a series of Monte Carlo runs using different entropy orders and batch-sizes. In the MonteCarlo runs, a random minimum-phase 15-tap FIR filter is chosen for the unknown channel impulse response, and the length of the deconvolving filter is set to that of the ideal inverse filter. For various values of  $N$  and  $\alpha$ , 100 random-choice (both for Cauchy distributed data samples and deconvolver initial weights) simulations are run for each combination of  $(N, \alpha)$ . The signal to interference ratio (SIR) of a single run is defined as the average of the SIR values of the last 100 iterations after convergence of that simulation (because due to the constant stepsize, the performance rattles slightly after convergence). The SIR value at a given iteration is computed as the ratio of the power of the maximum component of the overall filter to the power of the other components; that is, if we let  $a_n = h_n * w_n$  be the overall filter where  $w_n$  is the current estimate of the deconvolving filter, we evaluate

$$\text{SIR} = 10 \log_{10} \frac{[\max(a_i)]^2}{\sum_i a_i^2 - [\max(a_i)]^2}. \quad (8.49)$$

Note that under the assumption of wide sense stationary source signals, the power of the observed signal is time-invariant; therefore, the overall filter weights can equivalently be used to determine the signal-to-interference ratio. Regardless of the entropy order and batch size, an average deconvolution performance above 20 dB was attained [93]. As expected, with increased batch size, the SIR improved, although slightly. In addition, convergence was mostly achieved within 60 to 70 iterations (note that this number depends on the stepsize selected in the gradient descent algorithm), which makes this method potential interesting for practical applications.

## Blind Source Separation Using ITL

In the ICA literature, minimization of the mutual information between outputs (estimated source signals) is considered to be the natural information-theoretic “contrast function” which is the cost function in our terminology [47]. In spite of this understanding, two of the most well-known methods for ICA, (i.e., Bell and Sejnowski’s algorithm [26], and Hyvarinen’s FastICA [155]) use respectively the maximization of joint output entropy and negentropy. One difficulty in using Shannon’s MI is the estimation of the marginal entropies. In order to estimate the marginal entropy, Comon and others approximate the output marginal PDFs with truncated polynomial expansions [62], which naturally introduces error in the estimation procedure for densities far from the Gaussian. There are also parametric approaches to BSS, where the designer assumes a specific parametric model for the source distributions based on previous knowledge of the problem [59].

ITL algorithms are a natural fit for ICA because they manipulate either entropy or mutual information directly from data; that is, they are obvious competitors of the most widely applied algorithms utilized in the literature. We present below a review of three algorithms based on the maximization of the demixer joint output entropy, minimization of mutual information at the output of the demixer, and minimization of the sum of marginal entropies, however only the latter has been properly evaluated.

### Maximization of Output Entropy

Instead of using the Bell and Sejnowski learning rule in Eq. (8.17), Renyi’s quadratic entropy can implement the maximum entropy idea using a linear demixing network trained to maximize output entropy. We have used the MEE algorithm Eq. (3.16) to train the network to minimize the output information potential and the results in the data from the TIMIT database showed that this algorithm achieves similar signal-to-distortion ratio (SDR) to the Bell and Sejnowski algorithm (slightly higher if one of the sources is sub-Gaussian), and it is as simple to implement [251]. However, this method does not guarantee that only one source appears in each channel because the sum of marginal

entropies is used as the cost. Because it is difficult to measure effective separation of sources, for synthetic mixtures SDR is normally utilized and it is a slight modification of Eq. (8.49) defined as

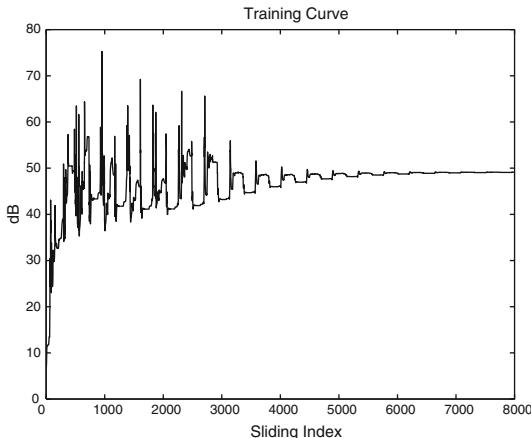
$$SDR = \frac{1}{n} \sum_{i=1}^n 10 \log_{10} \left( \frac{(\max q_i)^2}{q_i q_i^T - (\max q_i)^2} \right), \quad (8.50)$$

where  $\mathbf{Q} = \mathbf{RWH}$  is the overall processing matrix and  $q_i$  is the  $i$ th row of  $\mathbf{Q}$ . This criterion effectively measures the distance of  $\mathbf{Q}$  from an identity matrix, it is invariant to permutations and scaling, and it is a reasonable quality measure for source separation (apart from giving wrongly high values when two sources appear in the same channel, which is obviously detrimental, but most of the current algorithms avoid this condition).

### Minimization of Quadratic Mutual Information

The algorithm proposed by Xu et al. [339] avoids the polynomial expansion to approximate the marginal PDFs by using the nonparametric Parzen windowing with Gaussian kernels to estimate directly QMI<sub>ED</sub> or QMI<sub>CS</sub> at the output of the mapper as described in Chapter 2. Unfortunately, this method requires estimation of the joint output entropy, which is computationally expensive for many sources and requires in principle many samples for robust performance. Based on the generalized IP (GIP) criterion of Chapter 2, the demixing problem can be restated as finding a projection  $W$ , so that the GIP created by QMI is minimized. The advantage of the approach is that it can be applied to any linear or nonlinear demixing networks, because it is formulated as a cost function. As we have seen in Chapter 2, each of the terms in Eq. (2.104) can be computed with the information potential and so a nonparametric estimation of mutual information is achieved. Moreover multidimensional extensions have also been derived (see Eq. (2.105)).

One of the tests of this algorithm employed two speech signals from the TIMIT database as source signals. The mixing matrix is [1, 3.5; 0.8, 2.6] where two mixing directions [1, 3.5] and [0.8, 2.6] are similar. Whitening is first performed as a preprocessing stage. An online implementation is tried in this experiment, in which a short time window (200 samples) slides over the speech data (e.g., 10 samples/step). In each window position, the speech data within the window is used to calculate the GIP and the information forces are back-propagated using batch learning to adjust the demixing matrix. As the window slides at 10 samples/step the demixing matrix keeps being updated. The training curve (SDR vs. sliding index) is shown in Figure 8.16, which tells us that the method converges within 8000 samples of speech (0.5 sec) and achieves an SDR approaching 49.15 dB, which is comparable to other methods for this mixing condition [339]. The appeal of the method is its ability to track slowly varying mixing matrices (within 0.25 sec the demixing parameters are already close to their final positions). QMI<sub>CS</sub> can also be used and similar



**Fig. 8.16.** Training curve for BSS of two speech signals using a linear demixing network trained with QMI. Notice how quickly the SDR converges. SDR (dB) versus Iterations (from [252]).

results have been obtained. We experimentally verified that the result is not very sensitive to the kernel size in the cross-information potential estimation (a very large range of the kernel size will work, e.g., from 0.01 to 10).

Although the disadvantage of this method is the need to estimate the joint density to evaluate the mutual information that limits its applications to a small number of mixtures, the method efficiently uses the data (it can track changes in the demixing matrix) and can be immediately extended to non-linear mixtures by substituting the demixing system with an MLP. We have also verified experimentally that it is not necessary to have a good estimation of the joint entropy to obtain good separation performance, therefore QMI is a practical ICA algorithm for small number of sources.

### Minimization of Renyi's Mutual Information (MRMI)

The last algorithm for ICA to be reviewed here avoids the estimation of the joint PDF and was experimentally proven superior to many commonly accepted methods in extensive tests, in the sense that it requires fewer data to achieve the same performance level [148]. This algorithm works on whitened observations and uses only the sum of marginal Renyi's entropies estimated with the information potential.

It is well known that an instantaneous linear mixture can be separated by a spatial whitening (sphering) block followed by a pure rotation in  $N$  dimensions [62]. In fact, in general, for all BSS algorithms prewhitening is suggested to increase convergence speed [26]. The minimization of Renyi's mutual information (MRMI) algorithm already used in Chapter 6 for feature extraction, but for ICA we minimize Renyi's mutual information.

In this approach, the spatial (pre-) whitening matrix is evaluated from the observed data:  $\mathbf{W} = \mathbf{Q}\Lambda^{-1/2}$ , where  $\mathbf{Q}$  is the matrix of eigenvectors of the covariance matrix of the observations, and  $\Lambda$  is the corresponding eigenvalue matrix. Applying this transformation on the samples of the observation vector,  $\mathbf{x} = \mathbf{W}\mathbf{z}$ , we obtain the whitened samples  $\mathbf{x}$ . The rotation matrix that follows this whitening procedure is adapted according to a Renyi's inspired cost function to produce the outputs  $\mathbf{y} = \mathbf{R}(\theta)\mathbf{x}$ . Here,  $\theta$  denotes the set of Givens rotation angles that are used to parameterize the rotation matrix [148]. Now recall the following identity [148] that holds for an  $N$ -dimensional random vector  $\mathbf{y}$  and its marginals  $y^o$ , which relates the mutual information between the components of the random variable to the marginal entropies of these components and the joint entropy

$$I_S(Y) = \sum_{o=1}^N H_S(Y_o) - H_S(Y). \quad (8.51)$$

The same equality is not valid for Renyi's definitions of these quantities because Renyi's entropy lacks the recursivity property of Shannon's entropy, as was already stated in Chapter 6 for feature extraction. We show experimentally that a slightly modified Renyi's mutual information expression with the basic form of Eq. (8.51), given by

$$\sum_{o=1}^N H_\alpha(y_o) - H_\alpha(y) = \frac{1}{\alpha-1} \log \frac{\int_{-\infty}^{\infty} p_Y(y)^\alpha dy}{\int_{-\infty}^{\infty} \prod_{o=1}^n p_{Y^o}(y_o)^\alpha dy} \quad (8.52)$$

still can be used as a cost function in ICA with good results in many cases, provided we know or estimate the kurtosis of each source. Although Eq. (8.52) is not Renyi's mutual information, at the separating solution the criterion is guaranteed to be zero (the separating solution corresponds to letting the joint density, which is the integrand in the numerator, be equal to the product of marginal densities, which is the integrand in the denominator). But the converse does not hold (i.e., a value of zero does not imply independence, and when the sources are sub-Gaussians or have mixed kurtosis, small negative values of the cost are possible; see [150]).

Because only the rotation matrix in the separating topology is a function of the network parameters and Renyi's joint entropy is invariant to rotations (see Property 2.5), we can remove it and reduce the cost function to Eq. (8.53) which mimics the cost functions of Comon [62] with Renyi's entropy substituted for Shannon's,

$$J(\theta) = \sum_{o=1}^M H_\alpha(Y_o) \quad (8.53)$$

We [150] and Pham, et al. [246] showed that Eq. (8.53) is not a valid contrast for ICA in all cases, but we have also experimentally shown in [150] that it is possible to obtain separation results comparable to or better than other ICA methods with a slight modification to Eq. (8.53) when the sign of the kurtosis of the sources is taken into consideration as

$$J_\alpha(\theta) = \sum_{o=1}^M H_\alpha(Y_o) \text{sign} (E[Y_o^4 - 3Y_o^2]) \quad \alpha \geq 0, \quad \alpha \neq 1, \quad (8.54)$$

which is an extra calculation but can be readily done (for each value of  $\alpha$ , the kernel size parameter must be chosen appropriately; see [150]). In order to estimate the marginal entropies of each output  $y_o$ , we use the information potential for  $\alpha = 2$  and the SIG, yielding the cost function

$$\hat{J}_2(\theta) = - \sum_{o=1}^M \text{sign} \left( \sum_{k=1}^N (y_o^4(k) - 3y_o^2(k)) \right) \log \frac{1}{N} \sum_{k=1}^N G_{\sigma\sqrt{2}}(y_o(k) - y_o(k-1)) \quad (8.55)$$

The Givens rotation parameter vector  $\theta$  consists of  $M(M-1)/2$  parameters  $\theta_{ij}$ ,  $j > i$ , where each parameter represents the amount of Givens rotation in the corresponding  $i - j$  plane. The overall rotation matrix is the product of the individual in-plane rotation matrices

$$\mathbf{R}(\theta) = \prod_{i=1}^{M-1} \prod_{j=i+1}^M \mathbf{R}_{ij}(\theta_{ij}). \quad (8.56)$$

In Eq. (8.56), all products are performed sequentially from the right (or left). The important point is to perform these operations in the same order and from the same side when evaluating the gradient expression. The Givens rotation in the  $i - j$  plane is defined as an identity matrix whose  $(i, i)$ th,  $(i, j)$ th,  $(j, i)$ th, and  $(j, j)$ th entries, as in a rotation in two dimensions, are modified to read  $\cos \theta_{ij}$ ,  $-\sin \theta_{ij}$ ,  $\sin \theta_{ij}$ , and  $\cos \theta_{ij}$ , respectively.

## The MRMI Algorithm

The batch mode adaptation algorithm for the rotation matrix, which is parameterized in terms of Givens rotations, can be summarized as follows.

1. Whiten the observations  $\{z_1, \dots, z_N\}$  using  $\mathbf{W}$  to produce the samples  $\{x_1, \dots, x_N\}$ .
2. Initialize (randomly) the Givens rotation angles  $\theta_{ij}$ ,  $i = 1, \dots, n-1$ ,  $j = i+1, \dots, n$ .
3. Compute the rotation matrix using Eq. (8.56) and evaluate the output samples.
4. Until the algorithm converges repeat the following steepest descent procedure.

- a. Evaluate the gradient of the cost function  $J(\theta) = \sum_{o=1}^M \hat{H}_\alpha(Y_o)$ , using

$$\frac{\partial J}{\partial \theta_{ij}} = \sum_{o=1}^M \frac{\partial \hat{H}_\alpha(Y_o)}{\partial \theta_{ij}} = \sum_{o=1}^M \frac{1}{1-\alpha} \frac{\partial \hat{V}_\alpha(Y_o)/\partial \theta_{ij}}{\hat{V}_\alpha(Y_o)}, \quad (8.57)$$

where the information force  $\partial \hat{V}_\alpha / \partial y_o$  is estimated by Eq. (2.69) and

$$\mathbf{y}_{o,j} = \mathbf{R}^o \mathbf{x}_j, \quad o = 1, \dots, M, \quad j = 1, \dots, N$$

$$\frac{\partial y_{o,j}}{\partial \theta_{ij}} = \frac{\partial \mathbf{R}^o}{\partial \theta_{ij}} \mathbf{x}_j = \left( \frac{\partial \mathbf{R}}{\partial \theta_{ij}} \right)^o \mathbf{x}_j \quad (8.58)$$

$$\frac{\partial \mathbf{R}}{\partial \theta_{ij}} = \left( \prod_{p=1}^{i-1} \prod_{q=p+1}^M \mathbf{R}_{pq} \right) \left( \prod_{q=i}^{j-1} \mathbf{R}_{iq} \right) \\ \times \mathbf{R}'_{ij} \left( \prod_{q=j+1}^M \mathbf{R}_{iq} \right) \left( \prod_{p=i+1}^{M-1} \prod_{q=p+1}^M \mathbf{R}_{pq} \right), \quad (8.59)$$

where for any matrix  $\mathbf{A}$ ,  $\mathbf{A}^o$  denotes the  $o$ th row of that matrix and  $\mathbf{R}'_{ij}$  denotes the derivative of the specific Givens rotation matrix (in the  $i, j$  plane) with respect to its parameter  $\theta_{ij}$ .

- b. Evaluate the sign of the sum of kurtosis ( $K$ ), and update the Givens angles using

$$\theta_{ij} \leftarrow \theta_{ij} - \eta \operatorname{sign}(K) \frac{\partial J}{\partial \theta_{ij}}. \quad (8.60)$$

The algorithm above is for the separation of real-valued signals from real-valued mixtures. In order to generalize it to the case of complex-valued mixtures, the Givens matrices must be modified by incorporating imaginary parts to the rotation angles to account for rotations in the imaginary portions of the complex-valued vector space.

## Simulations for Batch MRMI

The whitening-rotation scheme has a very significant advantage. We observed experimentally that when this topology is used with a large number of samples, appropriately selected  $\alpha$ , and generalized Gaussian source distributions, there are no local minima of the cost function. Consider a two-source separation problem, for instance. The rotation matrix consists of a single parameter, which can assume values in the interval  $[0, 2\pi)$ . As far as separation is concerned, there are four equivalent solutions, which correspond to two permutations of the sources and the two possible signs for each source. The value of the cost function is periodic with  $\pi/2$  over the scalar rotation angle  $\theta$ , and most often is a very smooth function (sinusoidal like), which is easy to search using descent-based numerical optimization techniques.

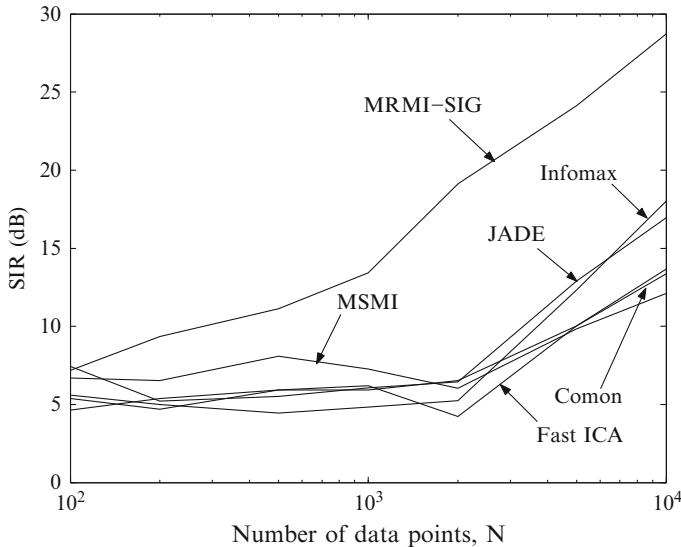
There are approximations in deriving Eq. (8.53), therefore we run numerous simulations using different  $\alpha$  and smooth kernels on synthetic and audio data instantaneous mixtures [148]. We first start with an investigation of the effect of  $\alpha$  on the separation of instantaneous mixtures, when the source kurtosis values span the range of super- and sub-Gaussian signals. In search of the answer to this question, a series of Monte Carlo simulations is performed (ten for each), using source distributions of different kurtosis values. In all these simulations, the two sources are assumed to have the same generalized Gaussian density, which is given by  $G_v(x) = C \cdot \exp(-|x|^v / (vE[|x|^v]))$ . The parameter  $v$  controls the kurtosis of the density and this family includes distributions ranging from Laplacian ( $v = 1$ ) to uniform ( $v \rightarrow \infty$ ). Gaussian distribution is a special case corresponding to ( $v = 2$ ), which leads to the classification of densities as super-Gaussian and sub-Gaussian for ( $v < 2$ ) and ( $v > 2$ ), respectively. For a given kurtosis value the training dataset is generated from the corresponding generalized Gaussian density and a random mixing matrix is selected. Then the separation is performed using various entropy orders (tracing the interval from 1.2 to 8 in steps of 0.4) and Gaussian kernels. The Gaussian kernel size was set at 0.25, and the adaptation using MRMI was run until a convergence of the SDR within a 0.1 dB band was achieved (although in practice this cannot be used as the stopping criterion), which usually occurred in less than 50 iterations with a stepsize of 0.2.

According to these simulations, the optimal entropy orders for the corresponding kurtosis value of the source densities are determined and are presented in Table 8.2. These results indicate that, for super-Gaussian sources, entropy orders greater than or equal to 2 should be preferred, whereas for sub-Gaussian sources, entropy orders smaller than 2, perhaps closer to 1 or even smaller than 1, should be preferred.

These results are in conformity to our expectations from the analysis of the information forces in Chapter 2. As we saw in that analysis, entropy orders larger than two emphasize samples in concentrated regions of data, whereas smaller orders emphasize the samples in sparse regions of data. If the mixtures belong to different kurtosis classes, then the quadratic entropy can be used as it puts equal emphasis on all data points regardless of their probability density. This effect is very different from some of the available algorithms where the

**Table 8.2.** Optimal Entropy Order Versus Source Density Kurtosis (From [150])

Kurtosis of Sources		Optimal Entropy Order $\alpha$
Super-Gaussian Sources	0.8 ( $v = 1$ )	6.4
	0.5 ( $v = 1.2$ )	5.2
	0.2 ( $v = 1.5$ )	2
	-0.8 ( $v = 4$ )	1.2
Sub-Gaussian Sources	-0.9 ( $v = 5$ )	1.6
	-1.0 ( $v = 6$ )	1.2



**Fig. 8.17.** SIR versus amount of data for the MRMI algorithm, Infomax, FICA, JADE, and MSMI, for the separation of five audio sources (from [148]).

BSS algorithms diverge if the source kurtosis is misestimated [26]. Another interesting aspect of this simulation is that it seems to imply that the Shannon information definition ( $\alpha \rightarrow 1$ ) is not particularly useful for separating super-Gaussian sources, although it might be useful for sub-Gaussian sources [150].

The next question addresses the performance of MRMI-SIG for a realistic source such as speech. Figure 8.17 depicts the SDR plots for the MRMI-SIG algorithm with  $\alpha = 2$  and Gaussian kernels, the FastICA (FICA) [155] with the symmetric approach and cubic nonlinearity, Infomax [26] with Amari, Cichocki, and Yang's natural gradient [5], and Comon's minimization of mutual information (MMI) using an instantaneous mixture of five audio sources [150]. The sources for each Monte Carlo trial were randomly drawn from one music piece, four female and five male speakers. Spatial prewhitening was used for each method, and the mixing matrix entries were chosen from a uniform density on  $[-1, 1]$ . The numbers in parentheses are the number of data samples used to train each algorithm. It is clearly seen from the figure that the MRMI-SIG method achieves better performance for a given amount of input data. The improved data efficiency of the MRMI-SIG method is discussed and shown in greater detail by Hild, et al. [150].

The MRMI-SIG method converges in fewer iterations than the others with a similar or lower computational complexity per update ( $O(N)$ ) than the other five methods. In the separation of temporally correlated sources such as speech, the MRMI-SIG shows a consistently better performance with respect to the Shannon entropy method (MSMI) implemented with kernel estimators, which become  $O(N^2)$ .

## 8.10 Conclusions

This chapter addresses algorithms for self-organization utilizing information theoretic concepts. The appeal of information theory stems from the ability of its descriptors (entropy and mutual information) to quantify the structure of the data. The appeal of self-organizing principles for unsupervised learning is that they provide “cost functions” that can be searched. The disadvantage is that most of the paradigms discovered thus far are still very much influenced by the ideas of information theory in a communication scenario, utilizing mutual information to quantify either channel capacity or rate distortion scenarios. We believe that there are many other information-theoretic principles for unsupervised and supervised learning to be discovered. We advance a novel principle, the principle of relevant information, which is able to create from a single cost function algorithms for all of the most commonly used applications of unsupervised learning (i.e., clustering, principal curves, and vector quantization). This principle generates automatically hierarchical features from the data, in fact searching a two-parameter space of a variational parameter and the kernel size. This can be very useful in particular if the learning machine has access to the operating point and can search meaningfully this space according to goals.

The other important aspect that this chapter discusses is the utility of the ITL nonparametric estimators of entropy and divergence to free the self-organizing principle of Infomax and I-Max from the conventional assumptions of linearity and Gaussianity that still permeate their application to real data. We have shown that simple algorithms to maximize output entropy resemble correlation learning (specifically the Hebbian rule), but they are more powerful than correlation learning as can be expected. In fact we showed that the eigendirections of PCA become curves that tend to pass through the data clusters. A practical application of minimization of entropy to blindly equalize a communication system was also presented.

Towards the end of the chapter we showed that the ITL cost function can be used for blind source separation, exploring not only the statistical but also the temporal characteristics of the data (when they posses time structure) when the stochastic gradient update is selected. The only algorithm’s that underwent sufficient testing in BSS are the MRMI and MRMI-SIG algorithms. They utilize the data very efficiently (i.e., achieve high SDR with small number of samples), which shows that they are quantifying higher-order statistics of the data better than the current methods. The MRMI-SIG is the core component of a fetal heart rate monitoring system [213] currently being developed for commercial use.

---

# A Reproducing Kernel Hilbert Space Framework for ITL

Jianwu Xu, Robert Jenssen, Antonio Paiva, and Il Park

## 9.1 Introduction

During the last decade, research on Mercer kernel-based learning algorithms has flourished [226, 289, 294]. These algorithms include, for example, the support vector machine (SVM) [63], kernel principal component analysis (KPCA) [289], and kernel Fisher discriminant analysis (KFDA) [219]. The common property of these methods is that they operate linearly, as they are explicitly expressed in terms of inner products in a transformed data space that is a reproducing kernel Hilbert space (RKHS). Most often they correspond to nonlinear operators in the data space, and they are still relatively easy to compute using the so-called “kernel-trick”. The kernel trick is no trick at all; it refers to a property of the RKHS that enables the computation of inner products in a potentially infinite-dimensional feature space, by a simple kernel evaluation in the input space. As we may expect, this is a computational saving step that is one of the big appeals of RKHS. At first glance one may even think that it defeats the “no free lunch theorem” (get something for nothing), but the fact of the matter is that the price of RKHS is the need for regularization and in the memory requirements as they are memory-intensive methods. Kernel-based methods (sometimes also called Mercer kernel methods) have been applied successfully in several applications, such as pattern and object recognition [194], time series prediction [225], and DNA and protein analysis [350], to name just a few.

Kernel-based methods rely on the assumption that projection to the high-dimensional feature space simplifies data handling as suggested by Cover’s theorem, who showed that the probability of shattering data (i.e., separating it exactly by a hyperplane) approaches one with a linear increase in space dimension [64]. In the case of the SVM, the assumption is that data classes become linearly separable, and therefore a separating hyperplane is sufficient for perfect classification. In practice, one cannot know for sure if this assumption holds. In fact, one has to hope that the user chooses a kernel (and its free

parameter) that shatters the data, and because this is improbable, the need to include the slack variable arises. The innovation of SVMs is exactly on how to train the classifiers with the principle of structural risk minimization [323].

ITL emerged independently of the research on Mercer kernel-based learning algorithms. In information-theoretic learning, the starting point is a dataset that globally conveys information about a real-world event. The goal is to capture the information in the parameters of a learning machine, using some information theoretic performance criterion. As we have seen, information-theoretic criteria are expressed as integrals over functions of probability densities. As a simplifying factor, ITL estimates the  $\alpha$ -norm of the PDF directly from data, without an explicit PDF estimation. Moreover, information-theoretic methods have the advantage over Mercer kernel-based methods in that they are easier to interpret.

In this chapter, we define bottom-up an RKHS for information-theoretic learning, named ITL RKHS, defined on the Hilbert space of square integrable PDFs. Then we provide a geometric perspective of all the ITL quantities presented in the previous chapters. Moreover, we show equivalences between Renyi's quadratic estimators of the statistical quantities and the Mercer kernel methods, which until now have been treated separately. Specifically, we show that Parzen window based estimators for Renyi's quadratic information measures have a dual interpretation as Mercer kernel-based measures, when they are expressed as functions of mean values in the Mercer kernel feature space. The Mercer kernel plays a similar role to the Parzen window of density estimation and they are shown to be equivalent. This means that if the Parzen window size can be reliably determined, then the corresponding Mercer kernel size is simultaneously determined by the same procedure.

Furthermore, we develop a classification rule based on the Euclidean distance between PDFs, and show that this corresponds to a linear classifier in the feature space. By regarding this classifier as a special case of the support vector machine, we provide an information theoretic interpretation of the SVM optimization criterion. This chapter is organized as follows. We start with the definition of the ITL RKHS and show the relation between the RKHS used in ITL and kernel methods. Then an ITL perspective of kernel learning and distances is explained, and a new information theoretic classification rule is derived. Thereafter, we analyze the connection between this classifier and the SVM and other kernel methods. The ITL RKHS structure offers an elegant and insightful geometric perspective towards information-theoretic learning and to the evaluation of statistics in kernel space. Finally, an application of RKHS to spike train analysis is presented to show the versatility of the approach in a difficult signal processing application.

## 9.2 A RKHS Framework for ITL

This section proposes a reproducing kernel Hilbert space (RKHS) framework for information-theoretic learning (ITL), not based on estimators but directly involving the PDFs. The issue of estimation from data samples is treated in Section 9.4.

The ITL RKHS is uniquely determined by the symmetric non-negative definite kernel function defined as *the cross information potential (CIP)* in ITL. The cross information potential between two PDFs  $p(x)$  and  $q(x)$ , defined in Chapter 2 as  $V(p, q) = \int p(x)q(x)dx$ , characterizes similarity between two stochastic functions. The information potential used so extensively in Chapters 3, 4, 5 and 6 as an entropic cost function (since it is the argument of the log of Renyi's quadratic entropy) is a special case obtained when  $p(x) = q(x)$ , i.e. a measure of self-similarity. CIP also appears both in the Euclidean and Cauchy-Schwarz divergence measures as well as in the QMICs and QMIED used in Chapters 6, 7 and 8 as a measure of dissimilarity in supervised and unsupervised learning.

We prove the existence of a congruence mapping between the ITL RKHS and the inner product space spanned by square integrable probability density functions. All the descriptors and cost functions in the original information-theoretic learning formulation can be re-written as algebraic computations on *deterministic functionals* in the ITL RKHS. We first focus on one-dimensional PDFs, and then consider the extension to multi-dimensions in Section 9.2. We form a  $L_2$  space spanned by all one-dimensional PDFs, and define an inner product in  $L_2$ . Since the inner product is symmetric non-negative definite, it uniquely determines the reproducing kernel Hilbert space for ITL which will be denoted as  $H_V$ . We then prove that the inner product itself is indeed a reproducing kernel in  $H_V$ .

### The $L_2$ Space of PDFs

Let  $E$  be the set that consists of all square integrable one-dimensional probability density functions over the real numbers; that is,  $f_i(x) \in E, \forall i \in I$ , where  $\int f_i(x)^2 dx < \infty$  and  $I$  is an index set. We then form a linear manifold  $\{\sum_{i \in K} \alpha_i f_i(x)\}$  for any  $K \subset I$  and  $\alpha_i \in R$ . We close the set topologically according to the convergence in the mean using the norm

$$\|f_i(x) - f_j(x)\| = \sqrt{\int (f_i(x) - f_j(x))^2 dx} \quad \forall i, j \in I \quad (9.1)$$

and denote the set of all linear combinations of PDFs and its limit points by  $L_2(E)$ .  $L_2(E)$  is an  $L_2$  space on PDFs. Moreover, by the theory of quadratically integrable functions, we know that the linear space  $L_2(E)$  forms a Hilbert

space if an inner product is imposed accordingly. Given any two PDFs  $f_i(x)$  and  $f_j(x)$  in  $E$ , we can define an inner product as

$$\langle f_i(x), f_j(x) \rangle_{L_2} = \int f_i(x) f_j(x) dx \quad \forall i, j \in I \quad (9.2)$$

Notice that this inner product is exactly the cross-information potential defined in Section 2.7. This definition of inner product has Eq. (9.1) as the corresponding norm. Hence,  $L_2(E)$  equipped with the inner product Eq. (9.2) is a Hilbert space. However, it is not a reproducing kernel Hilbert space because the inner product is not reproducing in  $L_2(E)$ : that is, the evaluation of any element in  $L_2(E)$  cannot be reproduced via the inner product between two functionals in  $L_2(E)$ . Next we show that the inner product of Eq. (9.2) is symmetric nonnegative definite, and by the Moore–Aronszajn theorem it uniquely determines the RKHS  $H_\nu$ .

### RKHS $H_\nu$ Based on $L_2(E)$

First, we define a bivariate function on the set  $E$  as

$$\nu(f_i, f_j) = \int f_i(x) f_j(x) dx \quad \forall i, j \in I \quad (9.3)$$

Even though  $\nu$  is defined on  $E$  its computation makes use of  $L_2(E)$ . However, by construction of  $L_2(E)$  as the span of  $E$ , any inner product defined on  $L_2(E)$  can be written as an inner product of elements of  $E$ . In reproducing kernel Hilbert space theory, the kernel function is a measure of similarity between functionals. Notice that Eq. (9.3) corresponds to the definition of the inner product in Eq. (9.2) and the cross-information potential between two PDFs, hence it is natural and meaningful to define the kernel function as  $\nu(f_i, f_j)$ . Next, we show that Eq. (9.3) is symmetric nonnegative definite in  $E$ .

**Property 9.1. (Non negative Definiteness):** The function in Eq. (9.3) is symmetric non negative definite in  $E \times E \rightarrow R$ .

*Proof.* The symmetry is obvious. Given any positive integer  $N$ , any set of  $\{f_1(x), f_2(x), \dots, f_N(x)\} \in E$  and any not all zero real numbers  $\{\alpha_1, \alpha_2, \dots, \alpha_N\}$ , by definition we have

$$\begin{aligned} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j \nu(f_i, f_j) &= \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j \int f_i(x) f_j(x) dx \\ &= \int \left( \sum_{i=1}^N \alpha_i f_i(x) \right) \left( \sum_{j=1}^N \alpha_j f_j(x) \right) dx = \int \left( \sum_{i=1}^N \alpha_i f_i(x) \right)^2 dx \geq 0. \end{aligned} \quad (9.4)$$

Hence,  $\nu(f_i, f_j)$  is symmetric non negative definite, and it is also a kernel function. According to the Moore–Aronszajn theorem [7, 222], there is a unique reproducing kernel Hilbert space, denoted by  $H_\nu$ , associated with the symmetric non negative definite function in Eq. (9.3). We construct the RKHS

$H_\nu$  bottom-up. Because this bivariate function is symmetric and non negative definite, it also has an eigen-decomposition by Mercer's theorem [217] as

$$\nu(f_i, f_j) = \sum_{k=1}^{\infty} \lambda_k \psi_k(f_i) \psi_k(f_j) \quad (9.5)$$

where  $\{\psi_k(f_i), k = 1, 2, \dots\}$  and  $\{\lambda_k, k = 1, 2, \dots\}$  are sequences of eigenfunctions and corresponding eigenvalues of the kernel function  $\nu(f_i, f_j)$ , respectively. The series above converges absolutely and uniformly on  $E \times E$ . Then we define a space  $H_\nu$  consisting of all functionals  $G(\cdot)$  whose evaluation for any given PDF  $f_i(x) \in E$  is defined as

$$G(f_i) = \sum_{k=1}^{\infty} \lambda_k a_k \psi_k(f_i), \quad (9.6)$$

where the sequence  $\{a_k, k = 1, 2, \dots\}$  satisfies the following condition

$$\sum_{k=1}^{\infty} \lambda_k a_k^2 < \infty. \quad (9.7)$$

Furthermore we define an inner product of two functionals in  $H_\nu$  as

$$\langle G, F \rangle_{H_\nu} = \sum_{k=1}^{\infty} \lambda_k a_k b_k, \quad (9.8)$$

where  $G$  and  $F$  are of form Eq. (9.6), and  $a_k$  and  $b_k$  satisfy property Eq. (9.7).

It can be verified that the space  $H_\nu$  equipped with the kernel function Eq. (9.3) is indeed a reproducing kernel Hilbert space and the kernel function  $\nu(f_i, \cdot)$  is a reproducing kernel because of the following two properties:

1.  $\nu(f_i, f_j)$  as a function of  $f_i(x)$  belongs to  $H_\nu$  for any given  $f_j(x) \in E$  because we can rewrite  $\nu(f_i, f_j)$  as

$$\nu(f_i, \cdot)(f_j) = \sum_{k=1}^{\infty} \lambda_k b_k \psi_k(f_j), \quad b_k = \psi_k(f_i)$$

That is, the constants  $\{b_k, k = 1, 2, \dots\}$  become the eigenfunctions  $\{\psi_k(f_i), k = 1, 2, \dots\}$  in the definition of  $G$ . Therefore,

$$\nu(f_i, \cdot) \in H_\nu, \quad \forall f_i(x) \in E \quad (9.9)$$

2. Given any  $G \in H_\nu$ , the inner product between the reproducing kernel and  $G$  yields the function itself by the definition Eq. (9.8),

$$\langle G, \nu(f_i, \cdot) \rangle_{H_\nu} = \sum_{k=0}^{\infty} \lambda_k a_k b_k = \sum_{k=0}^{\infty} \lambda_k a_k \psi_k(f_i) = G(f_i).$$

This is the so called reproducing property.

Therefore,  $H_\nu$  is a reproducing kernel Hilbert space with the kernel function and inner product defined above. By the reproducing property, we can rewrite the kernel function in Eq. (9.5) as

$$\boldsymbol{\nu}(f_i, f_j) = \langle \boldsymbol{\nu}(f_i, \cdot), \boldsymbol{\nu}(f_j, \cdot) \rangle_{H_\nu} \quad \boldsymbol{\nu}(f_i, \cdot) : f_i \mapsto \sqrt{\lambda_k} \psi_k(f_i), \quad k = 1, 2, \dots \quad (9.10)$$

The reproducing kernel nonlinearly maps the original PDF  $f_i(x)$  into the RKHS  $H_\nu$ .

We emphasize here that the reproducing kernel  $\boldsymbol{\nu}(f_i, f_j)$  is deterministic (due to the expected value in the inner product) and data-dependent in the sense that the mean of the norm of the nonlinearly transformed vector in  $H_\nu$  is dependent on the PDF of the original random variable because

$$\|\boldsymbol{\nu}(f_i, \cdot)\|^2 = \langle \boldsymbol{\nu}(f_i, \cdot), \boldsymbol{\nu}(f_i, \cdot) \rangle_{H_\nu} = \int f_i(x)^2 dx. \quad (9.11)$$

### Congruence Map Between $H_\nu$ and $L_2(E)$

We have presented two Hilbert spaces, the Hilbert space  $L_2(E)$  of PDFs and the reproducing kernel Hilbert space  $H_\nu$ . Even though their elements are very different, there actually exists a one-to-one congruence mapping  $\Psi$  (isometric isomorphism) from  $H_\nu$  onto  $L_2(E)$  such that

$$\Psi(\boldsymbol{\nu}(f_i, \cdot)) = f_i \quad (9.12)$$

Notice that the mapping  $\Psi$  preserves isometry between  $H_\nu$  and  $L_2(E)$  because by definitions of inner product Eq. (9.2) in  $L_2(E)$  and Eq. (9.10) in  $H_\nu$

$$\langle \boldsymbol{\nu}(f_i, \cdot), \boldsymbol{\nu}(f_j, \cdot) \rangle_{H_\nu} = \langle f_i(x), f_j(x) \rangle_{L_2} = \langle \Psi(\boldsymbol{\nu}(f_i, \cdot)), \Psi(\boldsymbol{\nu}(f_j, \cdot)) \rangle_{L_2}, \quad (9.13)$$

that is, the mapping  $\Psi$  maintains the inner products in both  $H_\nu$  and  $L_2(E)$ . In order to obtain an explicit representation of  $\Psi$ , we define an orthogonal function sequence  $\{\xi_m(x), m = 1, 2, \dots\}$  over the real numbers satisfying

$$\int \xi_k(x) \xi_m(x) dx = \begin{cases} 0, & k \neq m \\ \lambda_k, & k = m \end{cases} \quad \text{and} \quad \sum_{k=1}^{\infty} \psi_k(f_i) \int \xi_k(x) dx = 1, \quad (9.14)$$

where  $\{\lambda_k\}$  and  $\{\psi_k(f_i)\}$  are the eigenvalues and eigenfunctions evaluated at  $f_i$ , associated with the kernel function  $\boldsymbol{\nu}(f_i, f_j)$  by Mercer's theorem Eq. (9.5). We achieve an orthogonal decomposition of the probability density function as

$$f(x) = \sum_{k=1}^{\infty} \psi_k(f) \xi_k(x), \quad \forall f \in E. \quad (9.15)$$

The integration to unit of  $f$  is guaranteed by Eq. (9.14) (right). Note that the congruence map  $\Psi$  can be characterized as the unique mapping from  $H_\nu$  into

$L_2(E)$  satisfying the condition that for every functional  $G$  in  $H_\nu$  and every  $j$  in  $I$

$$\int \Psi(G) f_j(x) dx = \langle G, \boldsymbol{\nu}(f_j, \cdot) \rangle_{H_\nu} = G(f_j) \quad (9.16)$$

It is obvious that  $\Psi$  in Eq. (9.13) fulfills the condition Eq. (9.16). Then the congruence map can be represented explicitly as

$$\Psi(G) = \sum_{k=1}^{\infty} a_k \xi_k(x), \quad \forall G \in H_\nu, \quad (9.17)$$

where  $a_k$  satisfies condition Eq. (9.7). To prove the representation Eq. (9.17) is a valid and unique map, substituting Eq. (9.15) and Eq. (9.17) into Eq. (9.16), we obtain

$$\begin{aligned} & \int \sum_{k=1}^{\infty} a_k \xi_k(x) \sum_{m=1}^{\infty} \psi_m(f_j) \xi_m(x) dx \\ &= \sum_{k=1}^{\infty} \sum_{m=1}^{\infty} a_k \psi_m(f_j) \int \xi_k(x) \xi_m(x) dx \\ &= \sum_{k=1}^{\infty} \lambda_k a_k \psi_k(f_j) = G(f_j). \end{aligned} \quad (9.18)$$

In summary, we provide an explicit representation for the congruence map  $\Psi$  from  $H_\nu$  into  $L_2(E)$ . These two spaces are equivalent in this geometrical sense. However, it should be emphasized that the constituting elements are very different in nature. When using samples (realizations), the RKHS isometry framework offers a natural link between stochastic and deterministic functional analysis. Hence, it is more appealing to use  $H_\nu$  for information-theoretic learning, and we do not need the kernel trick.

## Extension to Multidimensional PDFs

Extension of  $H_\nu$  to multi-dimensional PDFs is straightforward because the definitions and derivations in the previous section can be easily adapted into multidimensional probability density functions. Now let  $E_m$  be the set that consists of all square integrable  $m$ -dimensional probability density functions, that is,  $f_{i,m}(x_1, \dots, x_m) \in E_m$ ,  $\forall i \in I$  and  $m \in N$ , where  $\int f_{i,m}(x_1, \dots, x_m)^2 dx_1, \dots, dx_m < \infty$  and  $I$  is the index set. We need to change the definition of kernel function Eq. (9.3) to

$$\boldsymbol{\nu}(f_{i,m}, f_{j,m}) = \int f_{i,m}(x_1, \dots, x_m) f_{j,m}(x_1, \dots, x_m) dx_1 \dots dx_m \quad \forall i, j \in I \quad (9.19)$$

Then every definition and derivation might as well be modified accordingly in the previous section. Let  $H_{\nu(m)}$  denote the reproducing kernel Hilbert space determined by the kernel function for  $m$ -dimensional PDFs. The proposed RKHS framework is consistent with dimensionality of PDFs.

The CIP based on the multidimensional PDFs characterizes the information among different random variables whose domains might not necessarily be the same in the whole space. In particular, the two-dimensional PDF CIP can be used to quantify the divergence or the cross-covariance between two random variables, because the joint PDF can be factorized into a product of two marginal PDFs as a special independent case. This is exactly on what the definitions of  $\text{QMI}_{\text{ED}}$  and  $\text{QMI}_{\text{CS}}$  in Chapter 2 are based. We use the two-dimensional PDF CIP to reformulate these two quantities in the following section.

### 9.3 ITL Cost Functions in the RKHS Framework

In this section, we re-examine the ITL cost functions introduced in Chapter 2 in the proposed ITL RKHS framework. First, as the kernel function  $\nu(f_i, f_j)$  in  $H_\nu$  is defined as the cross information potential between two PDFs, immediately we have

$$\int p(x)q(x)dx = \langle \nu(p, \cdot), \nu(q, \cdot) \rangle_{H_\nu}. \quad (9.20)$$

That is, the cross information potential is the inner product between two transformed functionals in  $H_\nu$ . The inner product quantifies similarity between two functionals which is consistent with the definition of cross-information potential. The information potential can thus be specified as the inner product of the functional with respect to itself

$$\int p(x)^2 dx = \langle \nu(p, \cdot), \nu(p, \cdot) \rangle_{H_\nu} = \|\nu(p, \cdot)\|_{H_\nu}^2. \quad (9.21)$$

The information potential appears as the norm square of the nonlinearly transformed functional in  $H_\nu$ . Therefore, minimizing error entropy in ITL turns out to be maximization of norm square in  $H_\nu$  (due to the minus sign in Renyi's quadratic entropy definition).

More interestingly, the result in Eq. (9.21) presents a new interpretation of Renyi's quadratic entropy. Because Renyi's quadratic entropy is the negative of the logarithm of the information potential, we obtain

$$H_2(X) = -\log \|\nu(p, \cdot)\|_{H_\nu}^2. \quad (9.22)$$

This means that there is an information theoretic interpretation for the log of the mean square of the transformed functional in  $H_\nu$ .

Based on the reformulations of cross information potential Eq. (9.18) and information potential Eq. (9.19) in  $H_\nu$ , we are ready to rewrite the 1-dimensional Euclidean and Cauchy–Schwarz distance measures in terms of operations on functionals in  $H_\nu$ . First,

$$D_{\text{ED}}(p, q) = \|\nu(p, \cdot) - \nu(q, \cdot)\|_{H_\nu}^2; \quad (9.23)$$

that is, the Euclidean distance measure is in fact the norm square of the difference between two corresponding functionals in  $H_\nu$ . The Cauchy-Schwarz divergence measure can be presented as

$$D_{CS}(p, q) = -\log \left( \frac{\langle \boldsymbol{\nu}(p, \cdot), \boldsymbol{\nu}(q, \cdot) \rangle_{H_\nu}}{\|\boldsymbol{\nu}(p, \cdot)\|_{H_\nu} \|\boldsymbol{\nu}(q, \cdot)\|_{H_\nu}} \right) = -\log(\cos \theta), \quad (9.24)$$

where  $\theta$  is the angle (in  $H_\nu$ ) between two functionals  $\boldsymbol{\nu}(p, \cdot)$  and  $\boldsymbol{\nu}(q, \cdot)$ . Therefore, the argument of the log of the Cauchy-Schwarz divergence measure truly depicts the separation of two functional vectors in  $H_\nu$ . When two vectors lie in the same direction the angle  $\theta = 0^\circ$  and  $D_{CS}(p, q) = 0$ . If two vectors are perpendicular to each other ( $\theta = 90^\circ$ ), and  $D_{CS}(p, q) = \infty$ . The RKHS  $H_\nu$  supplies rich geometric insights into the original definitions of the two divergence measures. Now we see that the geometric interpretation presented in Chapter 2, Section 2.7 is in fact accurate in  $H_\nu$ .

To extend the same formulation to the Euclidean and Cauchy-Schwarz quadratic mutual information defined in Section 2.7, consider the product of marginal PDFs  $f_1(x_1)f_2(x_2)$  as a special subset  $A_2$  of the 2-dimensional square integrable PDFs set  $E_2$  where the joint PDF can be factorized into product of marginals; that is,  $A_2 \subseteq E_2$ . Then both measures characterize different geometric information between the joint PDF and the factorized marginal PDFs. The Euclidean quadratic mutual information (QMI<sub>ED</sub>) can be expressed as

$$I_{ED}(X_1, X_2) = \|\boldsymbol{\nu}(f_{1,2}, \cdot) - \boldsymbol{\nu}(f_1 f_2, \cdot)\|_{H_V}^2, \quad (9.25)$$

where  $\boldsymbol{\nu}(f_{1,2}, \cdot)$  is the functional in  $H_{\nu(2)}$  corresponding to the joint PDF  $f_{1,2}(x_1, x_2)$ , and  $\boldsymbol{\nu}(f_1 f_2, \cdot)$  for the product of the marginal PDFs  $f_1(x_1)f_2(x_2)$ . Similarly, the Cauchy-Schwarz quadratic mutual information (QMI<sub>CS</sub>) can be rewritten as

$$I_{CS}(X_1, X_2) = -\log \frac{\langle \boldsymbol{\nu}(f_{1,2}, \cdot), \boldsymbol{\nu}(f_1 f_2, \cdot) \rangle_{H_V}}{\|\boldsymbol{\nu}(f_{1,2}, \cdot)\|_{H_V} \|\boldsymbol{\nu}(f_1 f_2, \cdot)\|_{H_V}} = -\log(\cos \gamma). \quad (9.26)$$

The angle  $\gamma$ , measured between  $f_{1,2}$  and  $f_1 f_2$  is the separation between two functional vectors in  $H_{\nu(2)}$ . When two random variables are independent ( $f_{1,2}(x_1, x_2) = f_1(x_1)f_2(x_2)$ ),  $\gamma = 0^\circ$  and the divergence measure is  $I_{CS}(f_1, f_2) = 0$  because two sets are equal. If  $\gamma = 90^\circ$ , two vectors in  $H_{\nu(2)}$  are orthogonal and the joint PDF is singular to the product of marginals. In this case, the divergence measure is infinity.

The proposed RKHS framework provides an elegant and insightful geometric perspective towards information-theoretic learning. All the ITL descriptors can now be re-expressed in terms of algebraic operations on functionals in  $H_\nu$ .

We can also provide a more mathematical understanding for the ITL operators and their properties. Let us start with the cross-information potential. From a statistical point of view, this quantity is a composite moment

(expected value over  $p(x)$  of another function  $q(x)$ ) of the r.v.  $x$ . In Chapter 7 we showed that it is a pseudo distance that was useful because it was easier to estimate than KL, Renyi's, MI, and other distance measures in probability spaces. Now we clearly see that it is the natural metric in  $H_\nu$ , because it defines the inner product in ITL RKHS. In the same RKHS we can define obviously other distances, such as  $D_{ED}$ ,  $D_{CS}$ , and  $QMI_{ED}$  and  $QMI_{CS}$  that are all dependent on the CIP. But now this picture becomes quite clear.

Another example is the evaluation of the statistical properties of IP. For instance, in Chapter 2 we say that the IP estimator contains higher-order statistical information of the input data. This has been recognized in ITL by applying the Taylor expansion to the Gaussian kernel used in the estimate of the information potential definition,

$$\hat{V}(X) = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \frac{1}{\sqrt{2\pi\sigma^2}} \sum_{k=0}^{\infty} \frac{1}{k!} \left[ -\frac{(x_i - x_j)^2}{2\sigma^2} \right]^k. \quad (9.27)$$

But notice that this result depends on the kernel utilized (the Gaussian kernel only provides sums of even-order moments, a polynomial kernel creates a finite sum of moments, etc.). From Eq. (9.21) it is clear that the norm maximization in ITL RKHS will include the PDF, therefore now we have a clean statement that derives from the use of the first order moment of the PDF in  $H_\nu$ .

## 9.4 ITL Estimators in RKHS

This section reinterprets the estimators of the information-theoretic quantities of the information potential and Euclidean and Cauchy–Schwarz distance in kernel spaces. Let  $H$  be a Hilbert space of real-valued functions defined on the real numbers  $R$ , equipped with an inner product  $\langle \cdot, \cdot \rangle$  and a real-valued non negative definite function  $\kappa(x, y)$  on  $R \times R$ . According to the Moore–Aronszajn theorem,  $\kappa(x, y)$  is a reproducing kernel, which means that each point in the input space  $R$  is mapped to a function in the RKHS defined by the selected kernel  $\kappa(\cdot, x) \in H_\kappa$ . We can define the nonlinear mapping between  $R$  and  $H_\kappa$  as  $\Phi(x) = \kappa(\cdot, x)$ , and obtain the reproducing property

$$\langle \Phi(x), \Phi(y) \rangle_{H_\kappa} = \langle \kappa(\cdot, x), \kappa(\cdot, y) \rangle = \kappa(x, y). \quad (9.28)$$

Therefore,  $\Phi(x) = \kappa(\cdot, x)$  defines the Hilbert space associated with the kernel. For our purposes here we will use the Gaussian kernel  $\kappa(x, y) = G_\sigma(x - y)$ , which is a nonnegative definite function, but many others can also be used. A Gaussian kernel corresponds to an infinite-dimensional Mercer kernel feature space, because the Gaussian has an infinite number of eigenfunctions.

This is very different from the reproducing kernel  $\nu(f_i, f_j)$  which has a norm dependent upon the PDF of the data as shown in Eq. (9.11). The norm

of the nonlinearly projected vector in the RKHS  $H_\kappa$  does not rely on the statistical information of the original data inasmuch as

$$\|\Phi(x)\|^2 = \langle \Phi(x), \Phi(x) \rangle_{H_k} = \kappa(0) \quad (9.29)$$

if we use translation-invariant kernel functions. Moreover, if  $x$  is a random variable,  $\Phi(x)$  is a function operating on random variable in  $H_\kappa$ . The value of  $\kappa(0)$  is a constant regardless of the original data. Consequently, the reproducing kernel Hilbert spaces  $H_\nu$  and  $H_\kappa$  determined by  $\nu(f_i, f_j)$  and  $\kappa(x, y)$ , respectively, are very different in nature, however, there are very interesting links among them as we show below.

### Estimator of the Information Potential

Recall the definition of the information potential *estimator* in Chapter 2, which is presented below for convenience,

$$\hat{V}(X) = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N G_{\sigma\sqrt{2}}(x_i - x_j).$$

Recall from Chapter 2 that this expression is obtained by integrating a product of two Gaussian functions centered at each sample over the domain which can be written as an inner product as

$$G_{\sigma\sqrt{2}}(x_i - x_j) = \kappa(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle_{H_k}. \quad (9.30)$$

Hence, the Parzen window-based estimator for the information potential can be expressed in terms of an inner product in the Mercer kernel space. We can further operate to obtain

$$\begin{aligned} \hat{V}(X) &= \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N G_{\sigma\sqrt{2}}(x_i - x_j) = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \langle \Phi(x_i), \Phi(x_j) \rangle_{H_k} \\ &= \left\langle \frac{1}{N} \sum_{j=1}^N \Phi(x_i), \frac{1}{N} \sum_{j=1}^N \Phi(x_i) \right\rangle_{H_k} = \mathbf{m}^T \mathbf{m} = \|\mathbf{m}\|^2. \end{aligned} \quad (9.31)$$

where  $N$  samples are assumed available from the data. That is, it turns out that the estimated information potential may be expressed as the squared norm of the mean vector of the data in a Mercer kernel feature space. This connection was previously pointed out in [112] in a study relating orthogonal series density estimation to kernel principal component analysis. Therefore the magic that may still have existed in why we could estimate an entropy without explicit PDF estimation becomes clear now. The argument of the log is a central moment of the projected data, therefore it is likely to find good estimators that do not require PDF estimation. As we mentioned in Chapter 2, estimating the mean and variance from data with the sample mean operator does not require PDF estimation, and the same applies to Renyi's quadratic entropy.

## Estimators of Quadratic Distances

In Chapter 2 we defined two distance measures using the information potential  $D_{ED}$  and  $D_{CS}$  and their corresponding estimators based on kernels. Let us assume that the goal is to estimate the distance between two PDFs  $p(x)$  and  $q(x)$  from which we have, respectively,  $N_1$  and  $N_2$  samples. Again for convenience we copy below the estimator for  $D_{ED}$ .

$$\hat{D}_{ED}(p, q) = \hat{V}_{ED} = \frac{1}{N_1^2} \sum_{i=1}^{N_1} \sum_{i'=1}^{N_1} G_{\sigma\sqrt{2}}(x_i - x_{i'}) - \frac{2}{N_1 N_2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} G_{\sigma\sqrt{2}}(x_i - x_j) + \frac{1}{N_2^2} \sum_{i=1}^{N_2} \sum_{j=1}^{N_2} G_{\sigma\sqrt{2}}(x_j - x_{j'}). \quad (9.32)$$

Note that we have for simplicity assumed that the same kernel size  $\sigma$  is appropriate for both estimators. This may not be the case in practice, but it can be incorporated in the subsequent analysis. In analogy to Eq. (9.32),  $D_{ED}$  may also be expressed in terms of mean vectors in the Mercer kernel feature space  $H_\kappa$  to obtain

$$\hat{D}_{ED}(p, q) = \|\mathbf{m}_1\|^2 - 2\mathbf{m}_1^T \mathbf{m}_2 + \|\mathbf{m}_2\|^2 = \|\mathbf{m}_1 - \mathbf{m}_2\|^2. \quad (9.33)$$

where  $\mathbf{m}_1$  is the kernel feature space mean vector of the data points drawn from  $p(x)$ , and  $\mathbf{m}_2$  is the kernel feature space mean vector of the data points drawn from  $q(x)$ . Hence,  $D_{ED}$  can also be seen to have a geometric interpretation in  $H_\kappa$ . It measures the square of the norm of the difference vector between the two means  $\mathbf{m}_1$  and  $\mathbf{m}_2$ . In a similar fashion we can obtain the estimator for the Cauchy–Schwarz divergence as

$$\hat{D}_{CS}(p, q) = -\log \left( \frac{\mathbf{m}_1^T \mathbf{m}_2}{\|\mathbf{m}_1\| \|\mathbf{m}_2\|} \right) = -\log (\cos \angle(\mathbf{m}_1, \mathbf{m}_2)). \quad (9.34)$$

Remember that the information cut explained in Chapter 6 was defined as the argument of the log of  $D_{CS}$ , therefore it has a dual interpretation as a measure of the cosine of the angle between cluster mean vectors in the Mercer kernel feature space  $H_\kappa$ . This metric is very natural in kernel machines because the nonlinear transformation induced by a symmetric reproducing kernel maps the input samples over a sphere in the feature space, because for any  $x$ ,  $\|\Phi(x)\|^2 = \kappa(0) = 1/(\sqrt{2\pi}\sigma)$ . Therefore, the distance between  $\Phi(x_i)$  and  $\Phi(x_j)$  on that sphere (i.e., the geodesic distance) is proportional to the angle between the vectors from the origin to those points

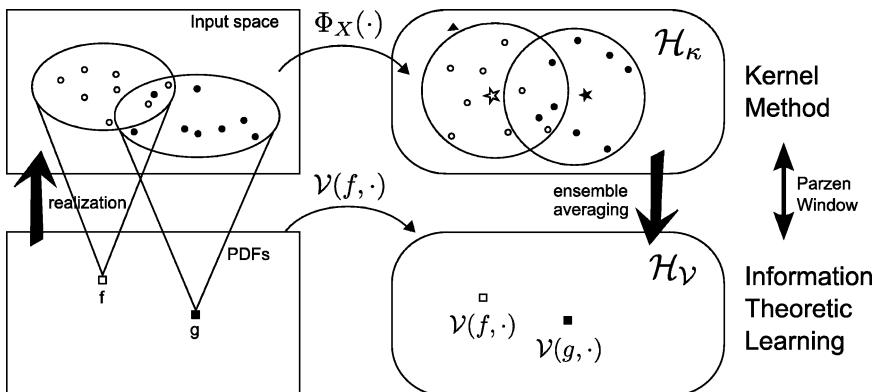
$$d(\Phi(x_i), \Phi(x_j)) \propto \cos^{-1} \left( \frac{\langle \Phi(x_i), \Phi(x_j) \rangle}{\|\Phi(x_i)\| \|\Phi(x_j)\|} \right) = \cos^{-1}(\sqrt{2\pi}\sigma\kappa(x_i - x_j)).$$

In other words, the kernel function is in fact computing the cosine of the angle between two points over the sphere (i.e., a distance). In addition, we

notice from the previous discussion that the transformed data  $\Phi(x_i)$  must lie on some embedded manifold over the positive hyperoctant on the sphere (because the kernel takes only positive values, the angle belongs to the interval  $[0, \pi/2]$ ).

The relationship between the two RKHS  $H_\nu$  and  $H_\kappa$  can then be readily perceived thru the ITL descriptors of  $IP$ ,  $D_{ED}$  and  $D_{CS}$  and their estimators. In fact, by comparing Eq. (9.22) with (9.31) (and Eq. (9.23) with (9.33) as well as Eq. (9.24) with (9.34)) we see that these statistical quantities in  $H_\nu$  can be estimated by the mean operator of the projected functionals in  $H_\kappa$ , which effectively were derived with Parzen's non-parametric asymptotically unbiased and consistent PDF estimator employed in the ITL cost functions. Provided one chooses a non negative definite kernel function as the Parzen window, the statistical quantities in  $H_\nu$  are related to the estimators in  $H_\kappa$  as illustrated in Figure 9.1.

In more general terms, the reproducing kernel Hilbert spaces  $H_\kappa$  and  $H_\nu$  are formally related via the expectation operator, which means that the feature map  $\nu(f_i, \cdot)$  is a transformation of the PDF space into a deterministic RKHS  $H_\nu$ , whereas the feature map  $\Phi(x)$  nonlinearly projects the sample space into a RKHS  $H_\kappa$  of functions operating on random variables. The feature map  $\nu(f_i, \cdot)$  is a descriptor of the stochasticity of the sample space, and immediate algebraic operation can be applied to compute statistics in  $H_\nu$ . This means that  $IP$ ,  $CIP$  and all the distances are deterministic scalars that quantify PDF properties. Hence, the proposed ITL RKHS framework provides a function analysis view of these statistics, and it seems the natural RKHS to perform statistical inference. Of course, the issue is that if one does not have an analytical description of the data PDF one can not progress further. This is where  $H_\kappa$  becomes useful because one can build there the estimators for the above mentioned statistical quantities. But this clearly shows that statistical estimators in  $H_\kappa$  operate with the full PDF information.



**Fig. 9.1.** The relationship between  $H_\nu$  and  $H_\kappa$  (from [341]).

Parzen was the first to introduce the RKHS methodology in statistical signal processing and time series analysis in the late 1950s. The essential idea is that there exists a congruence map between the RKHS of random variables spanned by the random process and its covariance function  $R(t, s) = E[X(t) X(s)]$  which determines a unique RKHS, denoted as  $H_R$ . This research line was briefly reviewed in Section 1.10. The conventional mean square error has also been rewritten as the norm square of projected vectors in  $H_R$  induced by the covariance function [238]. But  $H_R$  only takes the second-order statistics into account, whereas  $H_\nu$  is defined over PDFs and  $H_\kappa$ , depending upon the kernel utilized, will also implicitly embed all the statistical information of the data. Moreover, notice that  $H_\nu$  although linearly related to the PDF space, is nonlinearly related to the data space, unlike  $H_R$  which is linearly related to the data space. This implies that, with appropriate kernels,  $H_\kappa$  can be used to estimate second-and-higher order statistical information of the data. In general, mean and covariance operators are necessary to perform statistics in  $H_\kappa$  as we discuss next, but our work shows that the inclusion of the expected value operator in the kernel itself simplifies the analysis when the goal is statistical inference.

## 9.5 Connection Between ITL and Kernel Methods via RKHS $H_\nu$

In this section, we connect ITL and kernel methods via the proposed RKHS framework. As we have mentioned in the previous section, because  $H_\kappa$  is induced by the data-independent kernel function, the nonlinearly projected data in  $H_\kappa$  is still stochastic and statistical inference is required in order to compute quantities of interest. For instance, in order to compute the statistics over the functionals, the expectation and covariance operators are required. The expected value of functionals in  $H_\kappa$  is defined as  $E[\Phi(x)]$ . The cross-covariance is defined as a unique operator  $\Sigma_{\mathbf{XY}}$  such that for any functionals  $f$  and  $g$  in  $H_\kappa$

$$\left\langle g, \sum_{XY} f \right\rangle_{H_\kappa} = E[g(y)f(x)] - E[g(y)]E[f(x)] = \text{Cov}[f(x), g(y)]. \quad (9.35)$$

The mean and cross-covariance operators as statistics of functionals in  $H_\kappa$  become intermediate steps to compute other quantities such as the maximum mean discrepancy (MMD) [123], kernel independent component analysis (Kernel ICA) [27], and others. But the interesting question is to find out the relationships with both  $H_\nu$  and the ITL estimators of Equations (9.31), (9.33), and (9.34). We show here that MMD is equivalent to the Euclidean divergence measure, and that Kernel ICA is equivalent to the Cauchy–Schwarz quadratic mutual information. The statistical computations in  $H_\kappa$  have corresponding algebraic expressions in  $H_\nu$ .

## An ITL Perspective of Maximum Mean Discrepancy

The maximum mean discrepancy [126] is a statistical test based on kernel methods to determine whether two samples are from different distributions. Because the first-order moment of the PDF describes Renyi's quadratic entropy, theoretically, if the expected value of a PDF  $p(x)$  for an arbitrary measurable function is the same for both random variables, the two distributions are identical. Inasmuch as it is not practical to work with such a rich function class, MMD restricts the function class to a unit ball in a reproducing kernel Hilbert space  $H_\kappa$  that is associated with the kernel  $\kappa(\cdot, \cdot)$ . This leads to the following quantity,

$$M(X, Y) = \sup_{\|p\|_{H_\kappa} \leq 1} (E[p(x)] - E[p(y)]), \quad (9.36)$$

where  $X$  and  $Y$  are the underling random variables of the two distributions, and  $p$  is a family of measurable functionals in the unit ball of  $H_\kappa$ . The kernel trick can be employed here to compute MMD; that is,

$$p(x) = \langle \Phi(x), p \rangle_{H_\kappa} = \langle \kappa(x, \cdot), p \rangle_{H_\kappa}. \quad (9.37)$$

Substituting Eq. (9.37) into the definition of MMD Eq. (9.36), we obtain

$$M(X, Y) = \|m_X - m_Y\|_{H_\kappa}, \quad (9.38)$$

where  $m_X = E[\Phi(x)]$  and  $m_Y = E[\Phi(y)]$  are the statistical expectations of the functionals  $\Phi(x)$  and  $\Phi(y)$  in  $H_\kappa$ . Applying  $m_X = \frac{1}{N} \sum_{i=1}^N \Phi(x_i)$ , an empirical estimate of MMD can be obtained as

$$\left| \hat{M}(X, Y) \right|^2 = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \kappa(x_i, x_j) - \frac{2}{NL} \sum_{i=1}^N \sum_{j=1}^L \kappa(x_i, y_j) + \frac{1}{L^2} \sum_{i=1}^L \sum_{j=1}^l \kappa(y_i, y_j) \quad (9.39)$$

where  $\{x_i\}_{i=1}^N$  and  $\{y_j\}_{j=1}^L$  are two sets of data samples. The estimate of MMD provides a statistical test to determine whether two sets of data samples are from the same distribution. Comparing with Eq. (9.35), it is easy to show that MMD is equivalent to the Euclidean distance between PDFs ( $D_{ED}$ ); that is,  $[M(X, Y)]^2 = D_{ED}(f, g)$ . Moreover, because the Euclidean distance measure can be rewritten as the norm square difference between two functionals in  $H_\nu$ , we obtain

$$\|m_X - m_Y\|_{H_\kappa}^2 = \|\nu(f, \cdot) - \nu(g, \cdot)\|_{H_\kappa}^2. \quad (9.40)$$

The left-hand side is the norm square difference between two functional expectations in  $H_\kappa$ . The functional  $\Phi(x)$  is still stochastic in  $H_\kappa$ , thus the expectation operation is necessary to carry out the computation. On the other hand, the right-hand side is the norm square difference between two functionals in

the RKHS  $H_\nu$ . Because the functional  $\nu(f, \cdot)$  is deterministic, the computation is algebraic. The feature map  $\nu(f, \cdot)$  for  $H_\nu$  is equivalent to the expectation of the feature map  $\Phi(x)$  for  $H_\kappa$  (see Figure 9.1). Therefore, the proposed ITL RKHS framework provides a natural link between stochastic and deterministic functional analysis. The MMD in kernel methods is essentially equivalent to the Euclidean distance measure in information-theoretic learning.

### An ITL Perspective of Kernel ICA

Kernel ICA is a novel independent component analysis method based on a kernel measure of independence [12]. It assumes an  $H_\kappa$  determined by the kernel  $\kappa(x, y)$  and feature map  $\Phi(x)$ . The feature map  $\Phi(x)$  can be derived from the eigendecomposition of the kernel function  $\kappa(x, y)$  according to Mercer's theorem, and forms an orthogonal basis for  $H_\kappa$ . Then the  $F$ -correlation function is defined as the maximal correlation between the two random variables  $f_1(x_1)$  and  $f_2(x_2)$ , where  $f_1$  and  $f_2$  range over  $H_\kappa$ :

$$\rho = \max_{f_1, f_2} \text{corr}(f_1(x_1), f_2(x_2)) = \max_{f_1, f_2} \frac{\text{corr}(f_1(x_1), f_2(x_2))}{\sqrt{\text{var}(f_1(x_1))\text{var}(f_2(x_2))}}. \quad (9.41)$$

Obviously, if the random variables  $x_1$  and  $x_2$  are independent, then the  $F$ -correlation is zero. And the converse is also true provided that  $H_\kappa$  is large enough. This means that  $\rho = 0$  implies  $x_1$  and  $x_2$  are independent. In order to obtain a computationally tractable implementation of  $F$ -correlation, the reproducing property of RKHS is used to estimate the  $F$ -correlation. The nonlinear functionals  $f_1$  and  $f_2$  can be represented by the linear combination of the basis  $\{\Phi(x^i)\}_{i=1}^N$  in which  $\{x^i\}_{i=1}^N$  is an empirical observation of the random variable  $x$  with  $N$  samples. That is,

$$f_1 = \sum_{k=1}^N \alpha_1^k \Phi(x_1^k), \quad f_2 = \sum_{k=1}^N \alpha_2^k \Phi(x_2^k). \quad (9.42)$$

Substituting Eq. (9.42) and (9.37) into Eq. (9.41) and using the empirical data to approximate the population value, the  $F$ -correlation can be estimated as

$$\hat{\rho} = \max_{\alpha_1, \alpha_2} \frac{\alpha_1^T \mathbf{K}_1 \mathbf{K}_2 \alpha_2}{\sqrt{(\alpha_1^T \mathbf{K}_1^2 \alpha_1) (\alpha_2^T \mathbf{K}_2^2 \alpha_2)}} \quad (9.43)$$

where  $\mathbf{K}_1$  and  $\mathbf{K}_2$  are the Gram matrices associated with the datasets  $\{x_1^i\}_{i=1}^N$  and  $\{x_2^i\}_{i=1}^N$  defined as  $[K_i]_{a,b} = \kappa(x_i^a, x_i^b)$ .

Because the cost function in Eq. (9.43) is not a numerically stable estimator in general, a regularization is needed by penalizing the RKHS norms of  $f_1$  and  $f_2$  in the denominator of Eq. (9.43). The regularized estimator has the same

independence characterization property of the  $F$ -correlation, because it is the numerator,  $\alpha_1^T \mathbf{K}_1 \mathbf{K}_2 \alpha_2$ , that characterizes the independence property of two random variables.

We prove here the equivalence between the cost function used in Kernel ICA Eq. (9.43) and the Cauchy–Schwarz quadratic mutual information Eq. (9.34). To prove the equivalence, we use the weighted Parzen window which is defined as

$$\hat{f}(x) = \frac{1}{A} \sum_{i=1}^N \alpha_i \kappa(x, x_i), \quad (9.44)$$

where  $A$  is a normalization term such that the integral of  $\hat{f}(x)$  equals to 1.

When the Cauchy–Schwarz quadratic mutual information is used as a contrast function in ICA, it should be minimized so that the mutual information between random variables is also minimized. As the logarithm is a monotonic function, minimizing the Cauchy–Schwarz quadratic mutual information is equivalent to maximizing its argument. Therefore, by approximating the population expectation with sample mean for the argument in Eq. (9.34) and estimating the joint and marginal PDFs with weighed Parzen window Eq. (9.44), we obtain

$$\hat{J} = \max_{\alpha_1, \alpha_2} \frac{\alpha_1^T \mathbf{K}_1 \mathbf{K}_2 \alpha_2}{\sqrt{L (\mathbf{1}^T \mathbf{K}_1^2 \alpha_1) (\mathbf{1}^T \mathbf{K}_2^2 \alpha_2)}}, \quad (9.45)$$

where  $\mathbf{1} = [1, \dots, 1]^T$ ,  $[K_i]_{a,b} = \kappa(x_i^a, x_i^b)$ , and  $L = \sum_{i=1}^N \sum_{j=1}^N \alpha_1^i \kappa(x_1^i, x_1^j) \kappa(x_2^i, x_2^j) \alpha_2^j$ .

Comparing the two expressions Eq. (9.43) and (9.45), we notice that they have the same numerators but different normalizations. As we already pointed out, it is the numerators in the Kernel ICA and the Cauchy–Schwarz quadratic mutual information that characterize the dependence measure of two random variables. The denominators only provide normalization. Hence we conclude that the Cauchy–Schwarz quadratic mutual information, estimated via a weighted Parzen window, is equivalent to kernel ICA. Moreover, the coordinates of the nonlinear functionals  $f_1$  and  $f_2$  in the RKHS  $H_\kappa$  Eq. (9.42) have corresponding terms in the weighted Parzen window Eq. (9.44).

In summary, the feature map  $\Phi(x)$  works with individual data samples and transforms each datum into  $H_\kappa$  induced by the kernel  $\kappa(\cdot, \cdot)$ . For applications involving statistical inference on the transformed data, extra operators such as the mean and covariance are required. On the other hand, the feature map  $\nu(f, \cdot)$  deals with PDF directly and transforms each PDF into  $H_\nu$  determined by the kernel  $\nu(\cdot, \cdot)$ . If the applications are based on the statistics of the transformed functionals, only algebraic computation is needed without defining any extra operators as required in  $H_\kappa$ . Therefore the proposed RKHS framework provides a direct and elegant treatment of statistical inference using the RKHS technique. Certainly,  $H_\kappa$  is more flexible in other

applications beyond statistical inference inasmuch as it is based on the available data samples. The RKHS  $H_\nu$  is built directly upon PDFs, and requires Parzen windows to carry out the evaluation of the overall cost functions as we saw in ITL.

## 9.6 An ITL Perspective of MAP and SVM Classifiers

From the previous sections, we obtained a very clear view of the statistical power of mappings onto  $H_\kappa$ . The square of the projected data vector mean is in fact an estimator of the 2-norm of the PDF of the data as we saw in Chapter 2 when we interpreted quadratic Renyi's entropy. Therefore, a classification rule similar to LDA can be easily implemented with the projected class means without making any Gaussian assumption and benefiting from the high-dimensionality of the feature space that provides good performance even for linear classifiers. We will be interpreting functional as vectors in this perspective.

### Euclidean Distances in $H_\kappa$ and MAP Classifiers

The classification rule is based on  $D_{ED}$ , which we analyze theoretically both in the input space and in the Mercer kernel space  $H_\kappa$ . An interesting property of this classifier is that it contains the MAP classifier as a special case. We have available the training data points  $\{x_i\}; i = 1, \dots, N_1$ , drawn from  $p(x)$ , and a corresponding sample from  $q(x)$ ; that is,  $\{x_j\}; j = 1, \dots, N_2$ . The label information is used to create these two classes. Based on this training dataset we wish to construct a classifier which assigns a test data point  $x_0$  to one of the classes  $c_1$  or  $c_2$ . Now, we define

$$\hat{p}_o(x) = \frac{1}{N_1 + 1} \sum_{i=0}^{N_1} \kappa(x, x_i), \quad \hat{q}_o(x) = \frac{1}{N_2 + 1} \sum_{j=0}^{N_2} \kappa(x, x_j). \quad (9.46)$$

Hence,  $\hat{p}_o(x)$  is the Parzen estimator for  $p(x)$ , assuming  $x_0$  is included in the  $c_1$  data class. Likewise,  $\hat{q}_o(x)$  is the Parzen estimator for  $q(x)$ , assuming  $x_0$  is included in the  $c_2$  dataset. The proposed  $D_{ED}$ -based strategy is to classify  $x_0$  according to the following rule:

$$x_0 \in c_1 : \int (\hat{p}_o(x) - \hat{q}(x))^2 dx \geq \int (\hat{p}(x) - \hat{q}_o(x))^2 dx. \quad (9.47)$$

otherwise, assign  $x_0$  to  $c_2$ . In words, the rule assigns  $x_0$  to the class which, when having  $x_0$  appended to it, makes the estimated distance between the classes the greatest. We now analyze this simple classification rule in terms of the Mercer kernel feature space  $H_\kappa$ . Let  $\mathbf{m}'_i, i = 1, 2$  be the Mercer kernel feature space mean vector of class  $c_i$ , assuming  $\Phi(x_0)$  is assigned to that class. It is easily shown that

$$\begin{aligned}\mathbf{m}'_1 &= \frac{N_1}{N_1 + 1} \mathbf{m}_1 + \frac{1}{N_1 + 1} \Phi(x_0) \\ \mathbf{m}'_2 &= \frac{N_2}{N_2 + 1} \mathbf{m}_2 + \frac{1}{N_2 + 1} \Phi(x_0)\end{aligned}\quad (9.48)$$

In the kernel feature space, the equivalent classification rule of Eq. (9.47) may be expressed as

$$x_0 \in c_1: \|\mathbf{m}'_1 - \mathbf{m}_2\|^2 \geq \|\mathbf{m}_1 - \mathbf{m}'_2\|^2. \quad (9.49)$$

Assume that  $P(c_1) = P(c_2)$ , that is the prior probabilities for the classes are equal. Let  $P(c_1) = N_1/N$  and  $P(c_2) = N_2/N$ , which means that we assume  $N_1 = N_2$ . In this case,

$$\begin{aligned}\mathbf{m}'_1 &= \beta_1 \mathbf{m}_1 + \beta_2 \Phi(x_0) \\ \mathbf{m}'_2 &= \beta_1 \mathbf{m}_2 + \beta_2 \Phi(x_0)\end{aligned}\quad (9.50)$$

where  $\beta_1 = N_1/(N_1 + 1) = N_2/(N_2 + 1)$ , and  $\beta_2 = 1/(N_1 + 1) = 1/(N_2 + 1)$ . For ease of notation, let  $\Phi(x_0) = \mathbf{y}$ . The left-hand side of Eq. (9.49), becomes

$$\begin{aligned}\|\mathbf{m}'_1 - \mathbf{m}_2\|^2 &= \mathbf{m}'_1^T \mathbf{m}'_1 - 2\mathbf{m}'_1^T \mathbf{m}_2 + \mathbf{m}_2^T \mathbf{m}_2 \\ &= \beta_1^2 \|\mathbf{m}_1\|^2 + 2\beta_1\beta_2 \mathbf{m}_1^T \mathbf{y} + \beta_2^2 \|\mathbf{y}\|^2 - 2\beta_1 \mathbf{m}_1^T \mathbf{m}_2 \\ &\quad - 2\beta_2 \mathbf{m}_2^T \mathbf{y} + \|\mathbf{m}_2\|^2.\end{aligned}$$

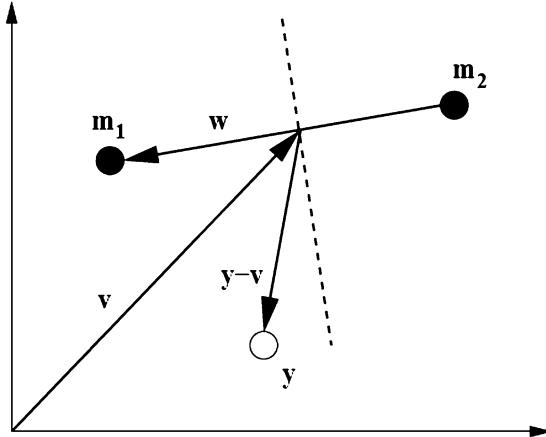
Similarly, the right-hand side of Eq.(9.46) becomes

$$\begin{aligned}\|\mathbf{m}_1 - \mathbf{m}'_2\|^2 &= \mathbf{m}_1^T \mathbf{m}_1 - 2\mathbf{m}_1^T \mathbf{m}'_2 + \mathbf{m}'_2^T \mathbf{m}_2 \\ &= \|\mathbf{m}_1\|^2 + 2\beta_1\beta_2 \mathbf{m}_2^T \mathbf{y} + \beta_2^2 \|\mathbf{y}\|^2 - 2\beta_1 \mathbf{m}_2^T \mathbf{m}_1 \\ &\quad - 2\beta_2 \mathbf{m}_1^T \mathbf{y} + \beta_1^2 \|\mathbf{m}_2\|^2.\end{aligned}$$

Using these results, the classification rule becomes

$$\begin{aligned}x_0 \in c_1: \quad &\|\mathbf{m}'_1 - \mathbf{m}_2\|^2 \geq \|\mathbf{m}_1 - \mathbf{m}'_2\|^2 \\ \Leftrightarrow &\mathbf{m}_1^T \mathbf{y} - \mathbf{m}_2^T \mathbf{y} - \frac{\beta_1^2 - 1}{2\beta_2(\beta_1 + 1)} (\|\mathbf{m}_2\|^2 - \|\mathbf{m}_1\|^2) \geq 0 \\ \Leftrightarrow &\mathbf{m}_1^T \mathbf{y} - \mathbf{m}_2^T \mathbf{y} + b \geq 0,\end{aligned}\quad (9.51)$$

where  $b = 1/2(\|\mathbf{m}_2\|^2 - \|\mathbf{m}_1\|^2)$ , and the constant  $(\beta_1^2 - 1)/\beta_2(\beta_1 + 1) = -1$ . The above classification rule has a simple geometrical interpretation. The point  $\mathbf{y}$  is assigned to the class whose mean it is closest, and the class boundary in kernel feature space is a hyperplane given by a vector  $\mathbf{w}$ . Let  $\mathbf{w} = \mathbf{m}_1 - \mathbf{m}_2$ , and let the midpoint between  $\mathbf{m}_1$  and  $\mathbf{m}_2$  be given by  $\mathbf{m}_c = 1/2(\mathbf{m}_1 + \mathbf{m}_2)$ . Now the class of  $\mathbf{y}$  is determined by examining whether the vector  $(\mathbf{y} - \mathbf{m}_c)$  encloses an angle smaller than  $\pi/2$  with the vector  $\mathbf{w}$ . If it does,  $\mathbf{y}$  is closest to  $\mathbf{m}_1$ , and  $\mathbf{y}$  is assigned to  $c_1$ .



**Fig. 9.2.** ISE-based geometric classification rule: assign the point  $y$  to the class to whose mean it is closest. This can be done by looking at the inner product between  $(y - m_c)$  and  $w$  ( $v$  in the figure is  $m_c$ ). It changes sign as the enclosed angle passes through  $\pi/2$ . The corresponding decision boundary is given by a hyperplane orthogonal to  $w$  (dashed line) (from [164]).

Hence,

$$\begin{aligned} x_0 \in c_1: \quad & \mathbf{w}^T(\mathbf{y} - \mathbf{m}_c) \geq 0 \\ & \mathbf{w}^T \mathbf{y} + b \geq 0 \\ \Leftrightarrow & \mathbf{m}_1^T \mathbf{y} - \mathbf{m}_2^T \mathbf{y} + b \geq 0. \end{aligned} \quad (9.52)$$

Figure 9.2 geometrically illustrates this simple classification rule, which we have derived using the DED criterion as a starting point. As explained above, in the Mercer kernel space, the value of the inner product between the class mean values and the new data point determines to which class it is assigned. The threshold value  $b$  depends on the squared Euclidean norms of the mean values, which are equivalent to the class information potentials, and hence the class entropies.

We now complete the circle, and analyze the Mercer kernel feature space classification rule in terms of Parzen estimators in the input space. Note that

$$\mathbf{m}_1^T \mathbf{y} = \mathbf{m}_1^T \Phi(x_0) = \frac{1}{N_1} \sum_{i=1}^{N_1} \Phi^T(x_i) \Phi(x_0) = \frac{1}{N_1} \sum_{i=1}^{N_1} \kappa(x_0, x_i) = \hat{p}(x_0). \quad (9.53)$$

Likewise

$$\mathbf{m}_2^T \mathbf{y} = \mathbf{m}_2^T \Phi(x_0) = \frac{1}{N_2} \sum_{j=1}^{N_2} \Phi^T(x_j) \Phi(x_0) = \frac{1}{N_2} \sum_{j=1}^{N_2} \kappa(x_0, x_j) = \hat{q}(x_0). \quad (9.54)$$

The classification rule hence becomes

$$x_0 \in c_1 : \quad \hat{p}(x_0) - \hat{q}(x_0) + b \geq 0. \quad (9.55)$$

We remark that this classification rule depends both on the estimated densities at  $x_0$ , and on the information potentials of the two classes. This forms the basis for the nonparameteric classifier explained in Chapter 6. In the case where the classes have the same value for the information potential (entropy), which means that the kernel feature space mean values have equal length from the origin, we have  $b = 0$ , and the current classification rule reduces to the well-known MAP classification rule (for equal priors), where the class probability densities are estimated using Parzen windowing. The same direct connection cannot be obtained based on the Cauchy–Schwarz divergence.

## The Support Vector Machine

The support vector machine is the most prominent Mercer kernel-based learning algorithm. It is a hyperplane classifier based on two crucial properties: (1) the kernel property (kernel trick), which allows for a relatively efficient computation of the SVM algorithm even in infinite dimensional spaces and, (2) the maximization of the hyperplane margin, which is a regularizing condition on the hyperplane solution. Basically, it limits the admissible separating hyperplanes to the one maximizing the margin. This regularization has a positive effect on the generalization capability of the classifier [323].

In the following, we give a brief review of the SVM theory. We formulate the problem directly in the Mercer kernel feature space. This Mercer kernel feature space is induced by some kernel function, which hopefully makes the feature space data linearly separable such that it can be separated by a hyperplane. Whether the data in fact are linearly separable, heavily depends on the user choosing a proper kernel.

Let  $c_1$  and  $c_2$  denote two data classes. We are given a training set consisting of  $\{x_i\}; i = 1, \dots, N_1$ , from  $c_1$ , and  $\{x_j\}; j = 1, \dots, N_2$ , from  $c_2$ . The task is to train a SVM classifier, such that it creates a maximum margin linear classifier in the kernel feature space. After training, the classification rule in feature space is

$$x_0 \in c_1 : \quad \mathbf{w}^{*T} \Phi(x_0) + b^* \geq 0. \quad (9.56)$$

otherwise,  $x_0 \in c_2$ , where  $x_0$  is a new, previously unseen data point. Presumably, it has either been generated by the process generating the  $c_1$  data, or the process generating the  $c_2$  data. Regularization by maximizing the margin in feature space corresponds to minimizing the squared norm of the (canonical) separating hyperplane weight vector, that is,  $\|\mathbf{w}^*\|^2$ , given the constraints

$$\begin{aligned} \mathbf{w}^{*T} \Phi(x_i) + b^* &\geq +1, & \forall x_i \in c_1 \\ \mathbf{w}^{*T} \Phi(x_j) + b^* &\leq -1, & \forall x_j \in c_2. \end{aligned} \quad (9.57)$$

This is a constrained optimization problem, which is dealt with by introducing Lagrange multipliers  $\alpha_i \geq 0, \alpha_j \geq 0$ , corresponding to the two classes, and a primal Lagrangian

$$L_P = \frac{1}{2} \|\mathbf{w}^*\|^2 - \sum_{i=1}^{N_1} \alpha_i \left[ \mathbf{w}^{*T} \Phi(x_i) + b^* - 1 \right] + \sum_{j=1}^{N_2} \alpha_j \left[ \mathbf{w}^{*T} \Phi(x_j) + b^* + 1 \right]. \quad (9.58)$$

The Lagrangian  $L_P$  has to be minimized with respect to the primal variables  $\mathbf{w}^*$  and  $b^*$ , and maximized with respect to the dual variables  $\alpha_i$  and  $\alpha_j$ . Hence, a saddle point must be found. At the saddle point, the derivatives of  $L_P$  with respect to the primal variables must vanish,

$$\frac{\partial L_p}{\partial b^*} = 0, \quad \frac{\partial L_p}{\partial \mathbf{w}^*} = 0, \quad (9.59)$$

which leads to

$$\sum_{i=1}^{N_1} \alpha_i = \sum_{j=1}^{N_2} \alpha_j = \Omega, \quad \mathbf{w}^* = \mathbf{m}_1^* - \mathbf{m}_2^*, \quad (9.60)$$

where

$$\mathbf{m}_1^* = \sum_{i=1}^{N_1} \alpha_i \Phi(x_i), \quad \mathbf{m}_2^* = \sum_{j=1}^{N_2} \alpha_j \Phi(x_j). \quad (9.61)$$

By substituting these constraints into Eq. (8.58), the dual Lagrangian

$$L_D = 2\Omega - \frac{1}{2} \left[ \sum_{i=1}^{N_1} \sum_{i'=1}^{N_1} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_{i'} \kappa(x_i, x_{i'}) - 2 \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j \kappa(x_i, x_j) \right. \\ \left. + \sum_{j=1}^{N_2} \sum_{j'=1}^{N_2} \boldsymbol{\alpha}_j \boldsymbol{\alpha}_{j'} \kappa(x_j, x_{j'}) \right] \quad (9.62)$$

is obtained, where  $\kappa(\cdot, \cdot)$  denotes an inner product between any two training data points in the Mercer kernel feature space  $H_\kappa$ .  $L_D$  must be maximized with respect to the Lagrange multipliers. It can be seen that the solution vector  $w^*$  has an expansion in terms of the training patterns weighted by the Lagrange multipliers. The Karush–Kuhn–Tucker (KKT) conditions

$$\begin{aligned} \alpha_i \left[ \mathbf{w}^{*T} \Phi(x_i) + b^* - 1 \right] &= 0, & \forall i = 1, \dots, N_1 \\ \alpha_j \left[ \mathbf{w}^{*T} \Phi(x_j) + b^* + 1 \right] &= 0, & \forall j = 1, \dots, N_2 \end{aligned} \quad (9.63)$$

specify the nonzero Lagrange multipliers to be those training patterns which are situated on the margin in feature space. Hence,  $\mathbf{w}^*$  is a weighted combination of the patterns on the margin. Let us determine the expression for

$b^*$  in the SVM theory. For those  $b^*$  corresponding to support vectors belonging to  $c_1$ , we have  $b_1^* = 1 - \mathbf{w}^{*T} \Phi(x_i)$ , where  $\Phi(x_i)$  is a support vector. By adding all  $b_1^*$  values corresponding to  $c_1$ , we have (remember that only those  $b_1^*$  are corresponding to support vectors deviate from zero)

$$\begin{aligned}\sum_{i=1}^{N_1} \alpha_i b_1^* &= \sum_{i=1}^{N_1} \alpha_i - \mathbf{w}^{*T} \sum_{i=1}^{N_1} \alpha_i \Phi(x_i) \\ \Omega b_1^* &= \Omega - \mathbf{w}^{*T} \mathbf{m}_1^* \\ b_1^* &= 1 - \frac{1}{\Omega} \|\mathbf{m}_1^*\|^2 + \frac{1}{\Omega} \mathbf{m}_1^{*T} \mathbf{m}_2^*. \end{aligned}\quad (9.64)$$

Similarly, for those  $b^*$  corresponding to support vectors belonging to  $c_2$ , we have  $b_2^* = -1 - \mathbf{w}^{*T} \Phi(x_j)$  and we obtain by adding them up

$$\begin{aligned}\sum_{j=1}^{N_2} \alpha_j b_2^* &= -\sum_{j=1}^{N_2} \alpha_j - \mathbf{w}^{*T} \sum_{j=1}^{N_2} \alpha_j \Phi(x_j) \\ \Omega b_2^* &= -\Omega - \mathbf{w}^{*T} \mathbf{m}_2^* \\ b_2^* &= -1 + \frac{1}{\Omega} \|\mathbf{m}_2^*\|^2 - \frac{1}{\Omega} \mathbf{m}_1^{*T} \mathbf{m}_2^*. \end{aligned}\quad (9.65)$$

Because  $b_1^* = b_2^*$ ,  $b^* = 1/2(b_1^* + b_2^*)$  which can be written as  $b^* = 1/\Omega(\|\mathbf{m}_2^*\|^2 - \|\mathbf{m}_1^*\|^2)$ .

### ITL Interpretation of the SVM

The classifier developed in the previous section is entirely determined by the mean vectors  $\mathbf{m}_1$  and  $\mathbf{m}_2$  of the training data, because both  $\mathbf{w}$  and  $b$  are determined by these vectors. For the classifier to perform well on test data, we are totally dependent on these mean vectors to truly represent the structure of the data. For example, the presence of outliers in the training set may affect the computation of  $\mathbf{w}$  and  $b$  in such a way that the performance of the classifier is degraded. This may be remedied by allowing the contribution of each training data point to the mean vectors to be weighted differently. Let us therefore introduce the weighting components  $\alpha_i > 0$  associated with  $c_1$ , and  $\alpha_j > 0$  associated with  $c_2$ . The weighted mean vectors then become

$$\mathbf{m}_1 = \frac{1}{\Omega_1} \sum_{i=1}^{N_1} \alpha_i \Phi(x_i), \quad \mathbf{m}_2 = \frac{1}{\Omega_2} \sum_{j=1}^{N_2} \alpha_j \Phi(x_j) \quad (9.66)$$

By introducing such weighted mean vectors, we also need to introduce some criterion to determine proper weights. Such a criterion should be optimal with respect to classifier performance. The performance of a classifier is measured by its success rate on test data. Hence, the classifier should generalize

well. In statistical learning theory, it has been shown that minimization of the squared norm of the hyperplane weight vector, although satisfying the classification constraints on the training data, improves generalization performance.

Based on the arguments above, we may relate the vector  $\mathbf{w} = \mathbf{m}_1 - \mathbf{m}_2$  to the SVM weight vector  $\mathbf{w}^* = \mathbf{m}_1^* - \mathbf{m}_2^*$ . Recall that the SVM is exactly based on regularization by minimization of  $\|\mathbf{w}^*\|^2$ . The minimization is accompanied by the classification constraints of Eq. (9.57), which ensures that the training data are classified correctly. Taking a closer look at the information potentials associated with the weighted mean vectors (Eq. (9.53)), we can write

$$\|\mathbf{m}_1\|^2 = \frac{1}{\Omega_1^2} \sum_{i=1}^{N_1} \sum_{i'=1}^{N_1} \alpha_i \alpha_{i'} \kappa(x_i, x_{i'}) = \int \hat{p}^2(x) dx. \quad (9.67)$$

Thus, the weighted mean vector  $\mathbf{m}_1$  is associated with

$$\hat{p}(x) = \frac{1}{\Omega_1} \sum_{i'=1}^{N_1} \alpha_i \kappa(x, x_{i'}), \quad (9.68)$$

a weighted Parzen window estimator in the input space. We likewise have the same for the second class. However, in this case, the kernels that constitute these Parzen window estimators are no longer equally important. Recall that to derive the original classification rule based on the  $D_{ED}$  of Eq. (9.55) we assumed that  $N_1 = N_2$ . Using the weighted Parzen window estimators instead, it is easily found that the corresponding assumption becomes  $\Omega_1 = \Omega_2 = \Omega$ . Therefore,

$$\mathbf{m}_1 = \frac{1}{\Omega} \mathbf{m}_1^*, \quad \mathbf{m}_2 = \frac{1}{\Omega} \mathbf{m}_2^*, \quad \mathbf{w} = \frac{1}{\Omega} \mathbf{w}^*. \quad (9.69)$$

Now, using the weighted Parzen window estimators we may express the SVM optimization problem in an information-theoretic framework as

$$\min_{\alpha_i, \alpha_j} \|\mathbf{w}^*\|^2 = \min_{\alpha_i, \alpha_j} \Omega^2 \|\mathbf{w}\|^2 = \min_{\alpha_i, \alpha_j} \Omega^2 \|\mathbf{m}_1 - \mathbf{m}_2\|^2 \quad (9.70)$$

because  $\|\mathbf{m}_1 - \mathbf{m}_2\|^2$  is the Mercer kernel feature space equivalent to  $D_{ED}$ , we have

$$\min_{\alpha_i, \alpha_j} \Omega^2 \|\mathbf{m}_1 - \mathbf{m}_2\|^2 = \min_{\alpha_i, \alpha_j} \Omega^2 \int (\hat{p}(x) - \hat{q}(x))^2 dx. \quad (9.71)$$

The optimization is subject to classification constraints, expressed as

$$\begin{aligned} \mathbf{w}^{*T} \Phi(x_i) + b^* &\geq +1 \Leftrightarrow \Omega \mathbf{w}^{*T} \Phi(x_i) + \Omega b \geq 1 \\ &\Leftrightarrow \hat{p}(x_i) - \hat{q}(x_i) + b \geq \frac{1}{\Omega} \quad i = 1, \dots, N_1 \end{aligned} \quad (9.72)$$

and

$$\begin{aligned} \mathbf{w}^{*T} \Phi(x_j) + b^* &\leq -1 \Leftrightarrow \Omega \mathbf{w}^{*T} \Phi(x_j) + \Omega b \leq -1 \\ &\Leftrightarrow \hat{p}(x_j) - \hat{q}(x_j) + b \leq -\frac{1}{\Omega} \quad j = 1, \dots, N_2. \end{aligned} \quad (9.73)$$

Therefore, the SVM classification rule, using the weighted Parzen window estimators, becomes

$$\begin{aligned} x_0 \in c_1 : \quad & \mathbf{w}^{*T} \Phi(x_j) + b^* \geq 0, \Leftrightarrow \Omega \mathbf{w}^T \Phi(x_0) + \Omega b \geq 0 \\ & \Leftrightarrow \hat{p}(x_0) - \hat{q}(x_0) + b \geq 0 \end{aligned} \quad (9.74)$$

The weighted Parzen window estimators  $\hat{p}(x)$ ,  $\hat{q}(x)$ , as defined above, are bona fide density estimators. That is, they are always non negative and integrate to one. However, because the weights are determined by minimizing  $D_{ED}$ , which puts emphasis on the points close to the class boundary trying to maximize the overlap between the class PDFs, we do not regard them as proper estimators for the PDFs that generated the data. From SVM theory, we know that in the Mercer kernel feature space, the only nonzero weighting components are those that correspond to data patterns on the margin.

In the input space, it seems that the corresponding nonzero weighting components will be associated with data patterns near the class boundary. We therefore interpret the minimization of the  $D_{ED}$  as a sparseness criterion, which tunes the classifier to those patterns near the boundary. The other data patterns should be much easier to classify correctly, and are not given any weight in the design of the classifier.

The performance of the classifier is secured by the classification constraints. Note that weighted Parzen window estimators have been previously proposed for improved Parzen window-based Bayes classification [11, 73]. In summary, we have found that one may view the SVM theory in feature space in terms of weighted Parzen density estimation in the input space, where regularization is obtained by minimizing the integrated squared error criterion. Hence, in an information-theoretic framework, the support vector machine is formulated by introducing the weights  $\alpha_i > 0$  and  $\alpha_j > 0$ , and estimating the class densities according to

$$\hat{p}(x) = \frac{1}{\Omega} \sum_{i=1}^{N_1} \boldsymbol{\alpha}_i \kappa(x, x_i), \quad \hat{q}(x) = \frac{1}{\Omega} \sum_{j=1}^{N_2} \boldsymbol{\alpha}_j \kappa(x, x_j). \quad (9.75)$$

The weights, and hence  $\hat{p}(x)$ ,  $\hat{q}(x)$ , are learned by enforcing a regularization criterion

$$\min_{\alpha_i, \alpha_j} \Omega^2 \int (\hat{p}(x) - \hat{q}(x))^2 dx \quad (9.76)$$

subject to the classification constraints,

$$\begin{aligned} \hat{p}(x_i) - \hat{q}(x_i) + b & \geq +\frac{1}{\Omega} \quad \forall x_i \in c_1 \\ \hat{p}(x_j) - \hat{q}(x_j) + b & \leq -\frac{1}{\Omega} \quad \forall x_j \in c_2 \end{aligned} \quad (9.77)$$

## 9.7 Case Study: RKHS for Computation with Spike Train

In the previous sections, we defined the ITL RKHS and showed several connections of this perspective with kernel methods. Although the original data space structure was sufficient to perform computations required for machine learning, the RKHS augmented this perspective and provided an elegant mathematical approach to information-theoretic learning. Certain types of data (e.g., point processes), however, do not naturally have the necessary structure to implement machine learning algorithms. For these cases, the structure can be obtained by the construction of an RKHS, and learning problems can easily be formulated and solved in the RKHS. As an application example we illustrate this methodology for spike trains. A spike train  $s \in S(T)$  is a simplified representation of a neuron's activity, specified by a sequence of ordered spike times  $s = \{t_m \in T : m = 1, \dots, N\}$  corresponding to the time instants in the interval  $T = [0, T]$  at which a neuron emits a spike (i.e., it “fires”) [67].

### Kernel Function for Spike Trains

The first step in the construction of the RKHS is to define a symmetric non negative definite kernel function. There are two basic approaches to do this for spike trains. The first approach follows the ideas from kernel methods, and utilizes the kernel  $\kappa$  to build an RKHS on spike times [48][290]. By writing the spike train as a sum of impulses centered at the spike times, and utilizing the linearity of the inner product in the RKHS, it is then possible to extend this RKHS to spike trains. An alternative approach is to build the RKHS by defining the kernel function on statistical descriptors of spike trains. We follow the latter approach because it is more insightful, and closely parallels the construction of the ITL RKHS.

A spike train is a realization of an underlying stochastic point process [307]. In general, to completely characterize a point process, the conditional intensity function must be used. However, for simplicity, here we focus on the special case of Poisson processes, which are memoryless and therefore the intensity function (or rate function) completely describes the point process [307]. The general case is considered in Paiva et al. [234]. In a sense, it can be said that intensity functions play for Poisson point processes the same role as PDFs for random variables, inasmuch as both are complete statistical functional descriptors [74]. Thus, it makes sense to build the RKHS for spike trains following an approach similar to the construction of the ITL RKHS.

Consider two spike trains,  $s_i, s_j \in S(T)$ , with  $i, j \in N$ . Denote the intensity of the underlying Poisson processes by  $\lambda_{s_i}(t)$  and  $\lambda_{s_j}(t)$ , respectively, where  $t \in [0, T]$  denotes the time coordinate. As with the CIP, we focus first on the case of deterministic statistical descriptors and consider the estimation problem in the next section. For any practical spike train with finite duration  $T$ , we have that

$$\int_T \lambda_{s_i}^2(t) dt < \infty. \quad (9.78)$$

As a consequence, the intensity functions of spike trains are valid elements of  $L_2([0, T])$ . Therefore, we can define in this space a kernel function of intensity functions given by the usual inner product in  $L_2$ ,

$$I(s_i, s_j) = \langle \lambda_{s_i}, \lambda_{s_j} \rangle_{L_2(T)} = \int_T \lambda_{s_i}(t) \lambda_{s_j}(t) dt. \quad (9.79)$$

We refer to  $I(\cdot, \cdot)$  as the memoryless cross-intensity (mCI) kernel. The proof that the mCI is indeed a symmetric non negative definite kernel, follows the same steps as the proof of Property 9.1, and is omitted here. Hence, the mCI induces an RKHS, denoted  $H_I$ . Comparing the definition of the mCI kernel with the CIP kernel, it is clear that both kernels incorporate the statistics descriptors directly into the kernel function. Thus, both are complete statistical operators. As with the CIP kernel, the definition of the mCI naturally induces a norm in the space of the intensity functions,

$$\|\lambda_{s_i}(\cdot)\|_{L_2(T)} = \sqrt{\langle \lambda_{s_i}, \lambda_{s_i} \rangle_{L_2(T)}} = \sqrt{\int_T \lambda_{s_i}^2(t) dt} \quad (9.80)$$

which is very useful for the formulation of optimization problems.

### Estimation of the Memoryless Cross-Intensity Kernel

Spike trains are realizations of underlying point processes, but, as defined, the mCI kernel is a deterministic operator on the point processes rather than on the observed spike trains. Thus, in practice, the kernel function is evaluated with the intensity functions estimated from spike trains. A well-known methodology for estimation of the intensity function is kernel smoothing [307]. Given a spike train  $s_i$  with spike times  $\{t_m^i \in T : m = 1, \dots, N_i\}$  the estimated intensity function is

$$\hat{\lambda}_{s_i}(t) = \sum_{m=1}^{N_i} h(t - t_m^i), \quad (9.81)$$

where  $h$  is the smoothing function. This function must be non-negative and integrate to one over the real line (just like a PDF). Commonly used smoothing functions are the Gaussian, Laplacian, and  $\alpha$ -functions, among others.

Consider spike trains  $s_i, s_j \in S(T)$  with estimated intensity functions  $\hat{\lambda}_{s_i}(t)$  and  $\hat{\lambda}_{s_j}(t)$ , according to Eq. (9.81). Substituting the estimated intensity functions in the definition of the mCI kernel (Eq. (9.79)) yields

$$\hat{I}(s_i, s_j) = \sum_{m=1}^{N_i} \sum_{n=1}^{N_j} \kappa(t_m^i - t_n^j). \quad (9.82)$$

where  $\kappa$  is the “kernel” obtained by the autocorrelation of the smoothing function  $h$ . The difference of the space of intensity functions from which the mCI kernel function was defined must be remarked, as well as the RKHS induced by this kernel. As with the CIP and the ITL RKHS, it can be shown there is a congruence mapping between the two spaces. Therefore, the same result can be obtained from either space.

It is interesting to verify the parallel of concepts and derived operators between the RKHS just defined for spike trains and the ITL RKHS. Yet, the most important result is that the construction of this RKHS provides the structure needed for computation with these data, which otherwise would not be possible.

## Principal Component Analysis

To exemplify these developments in an application, we now derive the algorithm to perform principal component analysis (PCA) of spike trains. The derivation of PCA in the RKHS is general, and applicable with other kernel functions. Interestingly, this is the traditional approach in the functional analysis literature [257]. A well-known example of discrete PCA done in an RKHS is kernel PCA [287, 288].

Consider a set of spike trains,  $\{s_i \in S(T), i = 1, \dots, N\}$ , for which we wish to determine the principal components. Computing the principal components of the spike trains directly is not feasible because we would not know how to define a principal component (PC), however, this is a trivial task in an RKHS. Let  $\{\Lambda_{s_i} \in H_I, i = 1, \dots, N\}$  be the set of elements in the RKHS corresponding to the given spike trains. Denote the mean of the transformed spike trains as

$$\bar{\Lambda} = \frac{1}{N} \sum_{i=1}^N \Lambda_{s_i}, \quad (9.83)$$

and the centered transformed spike trains (i.e., with the mean removed) can be obtained as

$$\tilde{\Lambda}_{s_i} = \Lambda_{s_i} - \bar{\Lambda}. \quad (9.84)$$

PCA finds an orthonormal transformation providing a compact description of the data. Determining the principal components of spike trains in the RKHS can be formulated as the problem of finding the set of orthonormal vectors in the RKHS such that the projections of the centered transformed spike trains  $\{\tilde{\Lambda}_{s_i}\}$  have maximum variance. This means that the principal components can be obtained by solving an optimization problem in the RKHS. A function  $\xi \in H_I$  (i.e.,  $\xi : S(T) \rightarrow R$ ) is a principal component if it maximizes the PCA cost function

$$J(\xi) = \sum_{i=1}^N \left[ \text{Proj}_\xi(\tilde{\Lambda}_{s_i}) \right]^2 - \rho (\|\xi\|^2 - 1), \quad (9.85)$$

where  $\text{Proj}_\xi(\tilde{\Lambda}_{s_i})$  denotes the projection of the  $i$ th centered transformed spike train onto  $\xi$ , and  $\rho$  is the Lagrange multiplier to the constraint  $(\|\xi\|^2 - 1)$  imposing that the principal components have unit norm. To evaluate this cost function one needs to be able to compute the projection and the norm of the principal components. However, the inner product needed for the projection and the norm are naturally defined. Thus, the above cost function can be expressed as

$$J(\xi) = \sum_{i=1}^N \left\langle \tilde{\Lambda}_{s_i}, \xi \right\rangle_{H_I}^2 - \rho (\langle \xi, \xi \rangle_{H_I} - 1), \quad (9.86)$$

By the representer theorem [288],  $\xi$  is restricted to the subspace spanned by the centered transformed spike trains  $\{\tilde{\Lambda}_{s_i}\}$ . Consequently, there exist coefficients  $b_1, \dots, b_N \in \mathbf{R}$  such that

$$\xi = \sum_{i=1}^N b_i \tilde{\Lambda}_{s_i} = b^T \tilde{\Lambda}, \quad (9.87)$$

where  $b^T = [b_1, \dots, b_N]$  and  $\tilde{\Lambda}(t) = [\tilde{\Lambda}_{s_1}(t), \dots, \tilde{\Lambda}_{s_N}(t)]^T$ . Substituting in Eq. (9.86) yields

$$\begin{aligned} J(\xi) &= \sum_{i=1}^N \left( \sum_{j=1}^N b_j \left\langle \tilde{\Lambda}_{s_i}, \tilde{\Lambda}_{s_j} \right\rangle \right) \left( \sum_{k=1}^N b_k \left\langle \tilde{\Lambda}_{s_i}, \tilde{\Lambda}_{s_k} \right\rangle \right) \\ &\quad + \rho \left( 1 - \sum_{j=1}^N \sum_{k=1}^N b_j b_k \left\langle \tilde{\Lambda}_{s_i}, \tilde{\Lambda}_{s_k} \right\rangle \right) \\ &= b^T \tilde{I}^2 b + \rho (1 - b^T \tilde{I} b). \end{aligned} \quad (9.88)$$

where  $\tilde{I}$  is the Gram matrix of the centered spike trains; that is, the  $N \times N$  matrix with elements

$$\begin{aligned} \tilde{I}_{ij} &= \left\langle \tilde{\Lambda}_{s_i}, \tilde{\Lambda}_{s_j} \right\rangle \\ &= \left\langle \Lambda_{s_i} - \bar{\Lambda}, \Lambda_{s_j} - \bar{\Lambda} \right\rangle \\ &= \langle \Lambda_{s_i}, \Lambda_{s_j} \rangle - \frac{1}{N} \sum_{l=1}^N \langle \Lambda_{s_i}, \Lambda_{s_l} \rangle - \frac{1}{N} \sum_{l=1}^N \langle \Lambda_{s_l}, \Lambda_{s_j} \rangle + \frac{1}{N^2} \sum_{l=1}^N \sum_{n=1}^N \langle \Lambda_{s_l}, \Lambda_{s_n} \rangle. \end{aligned} \quad (9.89)$$

In matrix notation,

$$\tilde{I} = I - \frac{1}{N} (1_N I + I 1_N) + \frac{1}{N^2} 1_N I 1_N, \quad (9.90)$$

where  $I$  is the Gram matrix of the inner product of spike trains  $I_{ij} = \langle \Lambda_{s_i}, \Lambda_{s_j} \rangle$ , and  $1_N$  is the  $N \times N$  matrix with all ones. This means that  $\tilde{I}$  can be computed directly in terms of  $I$  without the need to explicitly remove the mean of the transformed spike trains.

From Eq. (9.88), finding the principal components simplifies to the problem of estimating the coefficients  $\{b_i\}$  that maximize  $J(\xi)$ . Because  $J(\xi)$  is a quadratic function its extrema can be found by equating the gradient to zero. Taking the derivative with regard to  $b$  (which characterizes  $\xi$ ) and setting it to zero results in

$$\frac{\partial J(\xi)}{\partial b} = 2 \tilde{I}^2 b - 2\rho \tilde{I}b = 0, \quad (9.91)$$

and thus corresponds to the eigendecomposition problem

$$\tilde{I}b = \rho b. \quad (9.92)$$

This means that any eigenvector of the centered Gram matrix is a solution of Eq. (9.91). Thus, the eigenvectors determine the coefficients of Eq. (9.87) and characterize the principal components. It is easy to verify that, as expected, the variance of the projections onto each principal component equals the corresponding eigenvalue squared. So, the ordering of  $\rho$  specifies the relevance of the principal components.

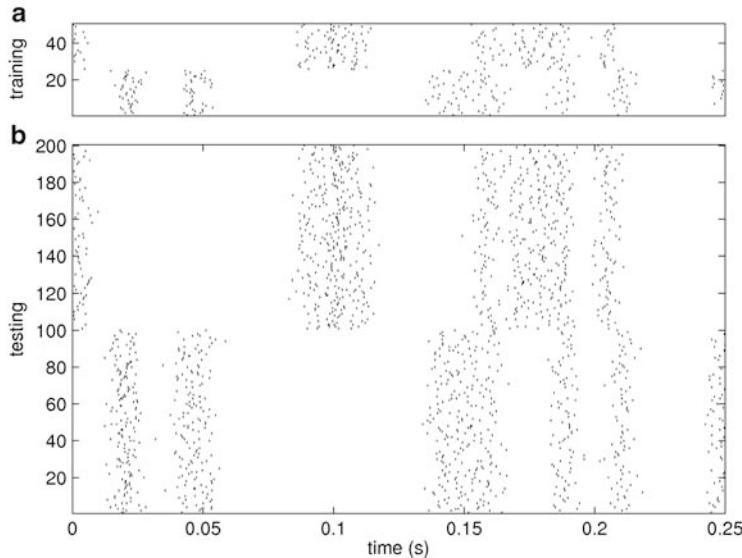
To compute the projection of a given input spike train  $s$  onto the  $k$ th principal component (corresponding to the eigenvector with the  $k$ th largest eigenvalue) we need only to compute in the RKHS the inner product of  $\Lambda_s$  with  $\xi_k$ . That is,

$$\begin{aligned} \text{Proj}_{\xi_k}(\Lambda_s) &= \langle \Lambda_s, \xi_k \rangle_{H_I} \\ &= \frac{1}{N} \sum_{i=1}^N b_{ki} \langle \Lambda_s, \tilde{\Lambda}_{s_i} \rangle \\ &= \frac{1}{N} \sum_{i=1}^N b_{ki} \left( I(s, s_i) - \frac{1}{N} \sum_{j=1}^N I(s, s_j) \right). \end{aligned} \quad (9.93)$$

An alternative approach to derive PCA for spike trains would be to utilize the inner product in the space of intensity functions directly. Basically, the derivation would follow the same steps but now in terms of intensity functions, rather than elements in the RKHS. Nevertheless, due to the congruence between this space and the RKHS induced by the mCI kernel, the result is the same. The key difference is that in Eq. (9.87), the principal components are written as weighted combinations of intensity functions, with weights given by the eigenvectors of the centered Gram matrix. That is, this approach allows the principal components to be obtained as intensity functions. Intensity functions characterize spike trains, therefore this perspective can be very telling of the underlying data structure.

## Examples with Synthetic Data

To illustrate the algorithm just derived we performed a simple experiment. We generated two template spike trains comprised of ten spikes uniformly random

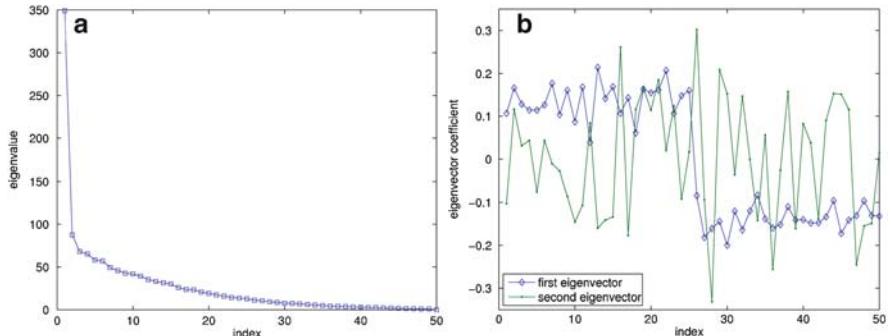


**Fig. 9.3.** (a) Spike trains used for evaluation of the eigendecomposition coefficients of PCA algorithm, and (b) for testing of the result. In either case, the first half of the spike trains corresponds to the first template and the remaining to the second template.

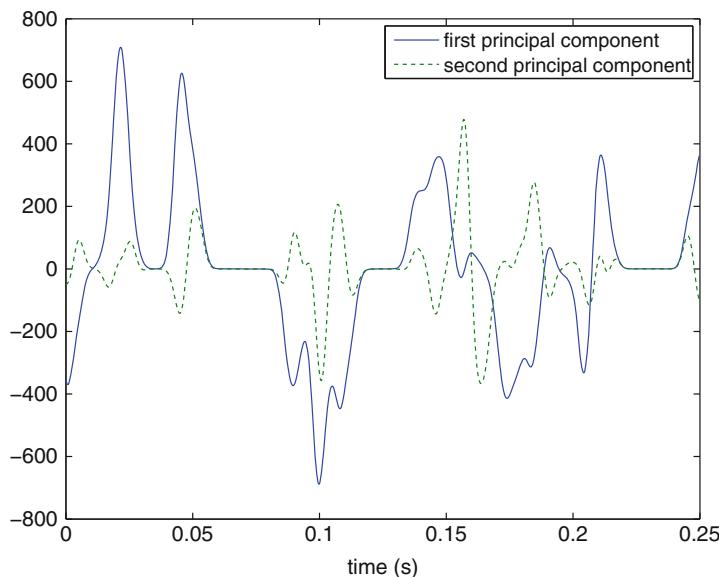
distributed over an interval of 0.25 sec. In a specific application these template spike trains could correspond, for example, to the average response of a culture of neurons to two distinct but fixed input stimuli. For the computation of the coefficients of the eigendecomposition (“training set”), we generated a total of 50 spike trains, half for each template, by randomly copying each spike from the template with probability 0.8 and adding zero mean Gaussian distributed jitter with standard deviation 3 ms. For testing of the obtained coefficients, 200 spike trains were generated following the same procedure. The simulated spike trains are shown in Figure 9.3.

With the PCA algorithm derived previously, we computed the eigendecomposition of the matrix  $\tilde{I}$ . The evaluation of the mCI kernel was estimated from the spike trains according to Eq. (9.82), and computed with a Gaussian kernel with size 2 ms. The eigenvalues  $\{\rho_l, l = 1, \dots, 100\}$  and first two eigenvectors are shown in Figure 9.4.

The first eigenvalue alone accounts for more than 26% of the variance of the dataset in the RKHS space. Although this value is not impressive, its importance is clear since it is nearly four times higher than the second eigenvalue (6.6%). Furthermore, notice that the first eigenvector clearly shows the separation between spike trains generated from different templates (Figure 9.4b). This again can be seen in the first principal component function, shown in



**Fig. 9.4.** Eigenvalues  $\{\rho_l, l = 1, \dots, 100\}$  in decreasing order (a) and first two eigenvectors (b) of the eigendecomposition of the centered Gram matrix  $\tilde{I}$ .

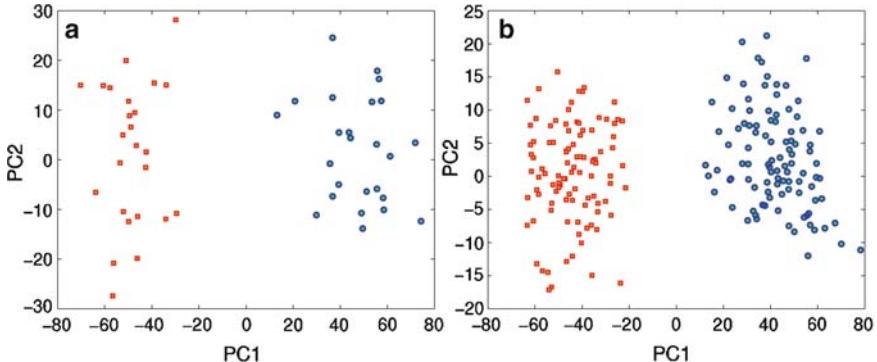


**Fig. 9.5.** First two principal component functions (i.e., eigenfunctions) in the space of intensity functions.

Figure 9.5, which reveals the location of the spike times used to generate the templates and discriminates between them with opposite signs.

Around periods of time where the spike from both templates overlap the first principal component is zero. As can be seen from the second principal component function, the role of the second eigenvector is to account for the dispersion in the data capable of differentiating spike trains generated from different templates.

For evaluation and testing, both datasets were projected onto the first two principal components. Figure 9.6 shows the projected spike trains. As noted



**Fig. 9.6.** Projection of spike trains onto the first two principal components. (a) Shows the projection of the spike trains in the training set and (b) in the testing set. The different point marks differentiate between spike trains corresponding to each one of the classes.

from the difference between the first and second eigenvalues, the first principal component is the main one responsible for the dispersion between classes of the projected spike trains. This happens because the direction of maximum variance is the one that passes through both clusters of points in the RKHS due to the small dispersion within class. The second principal component seems to be responsible for dispersion due to the jitter noise introduced in the spike trains, and suggests that other principal components play a similar role.

## 9.8 Conclusion

This chapter presented formally the relationship between the ITL descriptors in the preceding chapters and a RKHS that we called  $H_\nu$ . The elements of  $H_\nu$  are PDFs, and the kernel is the cross information potential. The inner product between two PDFs was crucial to measure distances for clustering and it also appeared in the divergence measures and quadratic mutual information. From the RKHS perspective we can easily recognize its central role because it defines the natural similarity metric in the space of PDFs.

In previous chapters we estimated all these quantities directly from samples using the information potential, that is the double sum of pairwise interactions between the data samples. We saw in this chapter that these estimators correspond rather directly to kernel methods. Indeed, when one places a kernel on a data sample we are defining a function that exists in a RKHS defined by the kernel. Therefore, we can reinterpret the information theoretical estimators as kernel operations: the information potential estimator is nothing but the mean square norm of the projected samples, the Cauchy–Schwarz divergence estimator is nothing but the log of the cosine of the angles between the projected samples.

Finally, we can establish a relationship between  $H_\nu$  and the  $H_\kappa$  defined by the kernel used in the ITL estimators: the mean value (over the data samples) in the RKHS defined by the kernel exists in the  $H_\nu$ . We were able to show that certain statistical operators defined in kernel spaces indeed correspond to ITL quantities (e.g., MDD becomes the Euclidean distance in  $H_\nu$ ).

It is therefore possible to also interpret well-known solutions in kernel methods with ITL descriptors. We showed that the SVM can be interpreted as the maximization of the Euclidean distance between classes estimated with a weighted Parzen window, where the weights are basically the Lagrange multipliers that define the support vectors.

We hope to have convinced the reader that the CIP  $V(p, q) = \int p(x)q(x)dx$  is the key concept behind all these relations, inasmuch as it defines  $H_\nu$ , and when  $p(x) = q(x)$  it defines the quadratic norm of the PDF that yields the estimators for Renyi's entropy. Therefore, synergisms were established between a statistical view of information and its functional view. The information-theoretical descriptors presented by Alfred Renyi just take the log of the projected data in  $H_\nu$ , which just changes the weighting of the inner product as we established in Chapter 2.

As an application of the RKHS methodology we apply it to a problem where the original space structure of the data does not support operations required for machine learning. One such example is the space of point processes. However, we can define a positive definite function in the point process space that builds a RKHS, where optimization algorithms can be carried out easily. We demonstrate this with PCA, but could likewise have used dissimilarity between spike trains, which as we have seen is associated with divergence.

# Correntropy for Random Variables: Properties and Applications in Statistical Inference

Weifeng Liu, Puskal Pokharel, Jianwu Xu, and Sohan Seth

## 10.1 Introduction

Similarity is a key concept to quantify temporal signals or static measurements. Similarity is difficult to define mathematically, however, one never really thinks too much about this difficulty and naturally translates similarity by correlation. This is one more example of how engrained second-order moment descriptors of the probability density function really are in scientific thinking. Successful engineering or pattern recognition solutions from these methodologies rely heavily on the Gaussianity and linearity assumptions, exactly for the same reasons discussed in Chapter 3.

We extend in this and the next chapter the fundamental definition of correlation for pairs of random variables and stationary random processes with a “generalized” correlation function called *correntropy* [201, 282] (the name origin is explained in Chapter 11). We briefly defined correntropy in Chapter 3 to show that the error entropy criterion is equivalent to robust least squares, but did not pursue it further. As correlation, correntropy is a bivariate function that produces a scalar, but it contains second and higher-order PDF moments, which are expressed by the kernel used in its definition. The higher moment description is not complete and does not appear in an explicit form, but as a sum of terms added to the value of the correlation weighted by the kernel size of ITL.

The reason we start with the case of two arbitrary random variables is to provide the probabilistic and geometric meaning of correntropy. Although the definition of correntropy is pretty general we have studied in detail the special case of symmetric translation-invariant kernels (e.g., Gaussian, Laplacian) and the results presented apply to the Gaussian kernel, denoted as  $G(\cdot, \cdot)$ . We show that correntropy is directly related to the probability of how similar two random variables are in a neighborhood of the joint space controlled by the kernel bandwidth; that is, the kernel bandwidth acts as a spotlight, controlling the “observation window” in which similarity is assessed. This adjustable window

provides an effective mechanism to eliminate the detrimental effect of outliers, and it is intrinsically different from the use of a threshold in conventional techniques.

Statistics carry a geometric meaning in sample space (e.g., MSE yields the 2-norm distance in sample space). We have shown in Chapter 3 that correntropy induces a new metric in sample space which is equivalent to the 2-norm distance if points are close, behaves similarly to the 1-norm distance as points get farther apart, and saturates as points are far apart approaching the zero-norm. This geometric interpretation elucidates the robustness of correntropy for outlier rejection and indeed implements Huber's M estimation.

## 10.2 Cross-Correntropy: Definitions and Properties

Consider two arbitrary scalar random variables  $X$  and  $Y$  defined over the real numbers.

**Definition.** *Cross-correntropy is a generalized similarity measure between two arbitrary scalar random variables  $X$  and  $Y$  defined by*

$$v(X, Y) = E_{XY}[\kappa(X, Y)] = \iint \kappa(x, y)p_{X,Y}(x, y)dxdy, \quad (10.1)$$

where the expected value is over the joint space and  $\kappa(\cdot, \cdot)$  is any continuous positive definite kernel. Cross-correntropy is a well-defined function provided  $\kappa(x, y)$  belongs to  $L_\infty$  (i.e. its maximal value is finite).

The conventional cross-correlation is a special case of cross-correntropy because the specific positive definite kernel  $\kappa(x, y) = xy$  substituted in Eq. (10.1) yields cross-correlation. We study in detail the special case of the Gaussian kernel (Eq. (1.50)) which is a symmetric, translation-invariant kernel. With this constraint Eq. (10.1) can be written

$$v_\sigma(X, Y) = E_{XY}[G_\sigma(X - Y)] = \iint G_\sigma(x - y)p_{X,Y}(x, y)dxdy, \quad (10.2)$$

where  $\sigma$  is the kernel size or bandwidth. The properties presented below generalizable with minor modifications to other symmetric translation-invariant kernels.

In practice, the joint PDF is unknown and only a finite number of data  $\{(x_i, y_i)\}_{i=1}^N$  are available, leading to the sample estimator of cross-correntropy

$$\hat{v}_{\sigma, N}(X, Y) = \frac{1}{N} \sum_{i=1}^N G_\sigma(x_i - y_i). \quad (10.3)$$

We can see an interesting link between cross-correntropy estimators and ITL estimators which are explored below. For simplicity, we refer to Eq. (10.1) or (10.2) as correntropy, and drop the explicit dependence on  $\sigma$ . Some important properties of correntropy as defined by Eq. (10.2) are presented below.

**Property 10.1.** For symmetric kernels, correntropy is symmetric; that is,  $v(X, Y) = v(Y, X)$ .

*Proof.* This property follows from the definitions of positive definiteness and symmetry.

**Property 10.2.** Correntropy is positive and bounded, and for the Gaussian kernel yields  $0 < v(X, Y) \leq 1/\sqrt{2\pi}\sigma$ . It reaches its maximum if and only if  $X = Y$ .

*Proof.* This can be easily verified with the definition of the Gaussian function.

**Property 10.3.** For the Gaussian kernel, correntropy is a weighted sum of all the even moments of the random variable  $Y-X$ .

*Proof.* Substituting the Taylor series expansion of the Gaussian function in Eq. (10.2), and assuming that it is valid to interchange the integral with the sum (all the moments of the PDF have to be finite), Eq. (10.2) yields

$$v_\sigma(X, Y) = \frac{1}{\sqrt{2\pi}\sigma} \sum_{n=0}^{\infty} \frac{(-1)^n}{2^n \sigma^{2n} n!} E[(X - Y)^{2n}]. \quad (10.4)$$

This is a very interesting expression that tells a lot about correntropy. First of all, one sees that for the Gaussian kernel, the sum of all even moments of the difference variable appear in correntropy. Thus correntropy keeps the nice bivariate form of correlation, but is still sensitive to the sum of second- and higher-order moments of the random variables. In many applications this sum may be sufficient to quantify better than correlation the relationships of interest and it is simpler to estimate than the higher-order moments. We do not recommend computing correntropy with Eq. (10.4).

The kernel size appears as a parameter weighting the second-order moment ( $n = 1$ ) and the higher-order moments. As  $\sigma$  increases above one, the high-order moments decay faster due to the denominator, so the second order moment tends to dominate and correntropy approaches (a biased) correlation. This has been verified in practice for kernel sizes 20 times larger than the variance of the data. Therefore, the issue of kernel size selection in correntropy is different from density estimation. As is demonstrated practically, the performance sensitivity of correntropy to the kernel size is much less than what could be expected from density estimation.

**Property 10.4.** When the Gaussian kernel shrinks to zero, correntropy approaches the value  $p(X = Y)$ ; that is,  $\lim_{\sigma \rightarrow 0} v_\sigma(X, Y) = \int p_{XY}(x, x) dx$ .

*Proof.* In the theory of distributions it can be proven that  $\lim_{\sigma \rightarrow 0} G_\sigma(x) \equiv \delta(x)$ . Therefore, for densities for which the integral exists, we can write

$$\begin{aligned}
\lim_{\sigma \rightarrow 0} v_\sigma(X, Y) &= \lim_{\sigma \rightarrow 0} \iint G_\sigma(x - y) p_{XY}(x, y) dx dy \\
&= \iint \delta(x - y) p_{XY}(x, y) dx dy \\
&= \int p_{XY}(x, x) dx
\end{aligned} \tag{10.5}$$

because by definition  $\delta(x - y) = 0$ ,  $(x - y) \neq 0$ . Eq. (10.5) represents a variable transformation of the joint PDF evaluated with equal arguments that yields a scalar value. Conceptually, we can interpret Eq. (10.5) as the integral over the domain of the intersection of the joint PDF with a plane passing through the origin with equation  $x = y$ . Hence, for the Gaussian kernel, the limit of correntropy for small kernel sizes approaches  $p(X = Y)$ . According to the conditions of the Parzen method [241], when  $\sigma$  goes to zero and the product  $N\sigma$  to infinity,  $\hat{p}_{X,Y,\sigma}(x, y)$  approaches  $p_{X,Y}(x, y)$  asymptotically in the mean square sense, therefore we can also expect a similar behavior for the estimated quantities.

**Property 10.4a.** Assume i.i.d. data  $\{(x_i, y_i)\}_{i=1}^N$  are drawn from the joint PDF  $p_{X,Y}(x, y)$ , and  $\hat{p}_{X,Y,\sigma}(x, y)$  is its Parzen estimate with kernel size  $\sigma$ . The correntropy estimate with kernel size  $\sigma\sqrt{2}$  is the integral of  $\hat{p}_{X,Y,\sigma}(x, y)$  along the line  $x = y$ ; that is,

$$\hat{v}_{\sigma\sqrt{2}}(X, Y) = \int \hat{p}_{X,Y,\sigma}(x, y)|_{x=y=u} du \tag{10.6}$$

*Proof.* Using the two dimensional radially symmetric Gaussian kernel to estimate the joint PDF, we have

$$\hat{p}_{X,Y,\sigma}(x, y) = \frac{1}{N} \sum_{i=1}^N G_\sigma \left( \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_i \\ y_i \end{bmatrix} \right), \tag{10.7}$$

where

$$G_\sigma \left( \begin{bmatrix} x \\ y \end{bmatrix} \right) = \frac{1}{2\pi|\Sigma|^{1/2}} \exp \left( -\frac{1}{2} \left( \begin{bmatrix} x \\ y \end{bmatrix}^T \Sigma^{-1} \begin{bmatrix} x \\ y \end{bmatrix} \right) \right), \quad \Sigma = \begin{pmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{pmatrix}$$

It is easy to see that with this choice

$$G_\sigma \left( \begin{bmatrix} x \\ y \end{bmatrix} \right) = G_\sigma(x)G_\sigma(y), \tag{10.8}$$

thus

$$\hat{p}_{X,Y,\sigma}(x, y) = \frac{1}{N} \sum_{i=1}^N G_\sigma(x - x_i)G_\sigma(y - y_i). \tag{10.9}$$

Integrating Eq. (10.9) along the line  $x = y$  and assuming the integral exists, we obtain

$$\begin{aligned}
\int_{-\infty}^{\infty} \hat{p}_{X,Y,\sigma}(x,y)|_{x=y=u} du &= \int_{-\infty}^{\infty} \frac{1}{N} \sum_{i=1}^N G_{\sigma}(x - x_i) G_{\sigma}(y - y_i) |_{x=y=u} du \\
&= \frac{1}{N} \sum_{i=1}^N \int_{-\infty}^{\infty} G_{\sigma}(u - x_i) G_{\sigma}(u - y_i) du \\
&= \frac{1}{N} \sum_{i=1}^N G_{\sigma\sqrt{2}}(x_i - y_i) = \hat{v}_{\sigma\sqrt{2}}(X, Y). \quad (10.10)
\end{aligned}$$

This completes the proof.

In practical applications, the estimation of correntropy is done only with a finite number of samples, which sets a lower bound on the kernel size, because too small a kernel size will lead to meaningless estimation. When the kernel size used in correntropy is  $\sigma$ , its rectangle approximation has a bandwidth  $\sqrt{\pi/2}\sigma$  and assume further that the joint PDF is smooth in this bandwidth. Then a more precise interpretation of correntropy with Gaussian kernels is

$$v_{\sigma}(X - Y) \approx p(|Y - X| < \sqrt{\pi/2}\sigma) / \sqrt{2\pi}\sigma. \quad (10.11)$$

Let us define the random variable  $E = X - Y$  with PDF  $p(e)$ .

**Property 10.5.** When the Gaussian kernel shrinks to zero,  $v_{\sigma}(X, Y)$  is the value of  $p_E(e)$  evaluated at  $e = 0$ ; i.e.,  $\lim_{\sigma \rightarrow 0} v_{\sigma}(X, Y) = p_E(0)$

*Proof.* In fact,

$$v_{\sigma}(X, Y) = E_{XY}[G_{\sigma}(X - Y)] = E[G_{\sigma}(E)] = \int G_{\sigma}(e)p(e)de, \quad (10.12)$$

and

$$p_E(0) = p(X = Y) = \int p_{X,Y}(x, x)dx, \quad (10.13)$$

therefore the proof follows immediately from Property 10.4. There is an equivalent statement in terms of estimators.

**Property 10.5a.** Assume the samples  $\{(x_i, y_i)\}_{i=1}^N$  are drawn from the joint PDF  $p_{X,Y}(x, y)$ . Define the error random variable  $E = Y - X$ , and  $\hat{p}_{E,\sigma}(e)$  as the Parzen estimate of the error PDF from data  $\{(e_i = x_i - y_i)\}_{i=1}^N$ . Then  $\hat{v}_{\sigma}(X, Y)$  is the value of  $\hat{p}_{E,\sigma}(e)$  evaluated at the point  $e = 0$ ; that is,

$$\hat{v}_{\sigma}(X, Y) = \hat{p}_{E,\sigma}(0). \quad (10.14)$$

*Proof.* By Eq. (10.2)

$$\hat{v}_{\sigma}(X, Y) = \frac{1}{N} \sum_{i=1}^N G_{\sigma}(x_i - y_i) = \frac{1}{N} \sum_{i=1}^N G_{\sigma}(e_i) = \hat{p}_{E,\sigma}(0), \quad (10.15)$$

which completes the proof.

**Property 10.6.** Under the condition  $N \rightarrow \infty$ ,  $\hat{v}_{N,\sigma}(X, Y)$  is an estimator of  $v_\sigma(X, Y)$  consistent in mean square. Furthermore, under the conditions  $N\sigma \rightarrow \infty$  and  $\sigma \rightarrow 0$ ,  $\hat{v}_{N,\sigma}(X, Y)$  is an asymptotically unbiased estimator of  $p_E(0)$  and consistent in mean square.

*Proof.* The proof follows from the properties of Parzen estimation [241]. Indeed,

$$\mathbb{E}[\hat{v}_{N,\sigma}(X, Y)] = v_\sigma(X, Y) \quad (10.16)$$

$$\lim_{N \rightarrow \infty, \sigma \rightarrow 0} \mathbb{E}[\hat{v}_{N,\sigma}(X, Y)] = p_E(0) \quad (10.17)$$

$$\text{var}[\hat{v}_{N,\sigma}(X, Y)] = N^{-1} \text{var}[G_\sigma(E)] \quad (10.18)$$

$$\lim_{N \rightarrow \infty, \sigma \rightarrow 0} N\sigma \text{var}[\hat{v}_{N,\sigma}(X, Y)] = p_E(0) \int G_1^2(u) du, \quad (10.19)$$

where  $G_1(u)$  is the Gaussian kernel with  $\sigma = 1$ .

From these properties we can conclude that the effect of the kernel size  $\sigma$  in correntropy is not as dramatic as in ITL. In fact, let us assume the error PDF is Gaussian with standard deviation  $\sigma_E$ . Then from Eq. (10.11)

$$v_\sigma(X, Y) = 1/\sqrt{2\pi(\sigma^2 + \sigma_E^2)} \quad \text{and} \quad v_0 =: \lim_{\sigma \rightarrow 0} v_\sigma = 1/\sqrt{2\pi}\sigma_E. \quad (10.20)$$

A simple calculation shows that if  $\sigma < 0.32\sigma_E$ ,  $|v_\sigma - v_0|/v_0 < 0.05$ . For  $\sigma^2 \ll \sigma_E^2$ , we obtain  $(v_\sigma - v_0)/v_0 \approx \sigma^2/(2\sigma_E^2)$ . More generally, taking the derivative of the correntropy w.r.t.  $\sigma$ ,

$$\left| \frac{dv_\sigma(X, Y)}{d\sigma} \right| = \left| \frac{\sigma}{(\sigma^2 + \sigma_E^2) \sqrt{2\pi(\sigma^2 + \sigma_E^2)}} \right| < \frac{1}{\sqrt{2\pi}\sigma_E^2}$$

Therefore, if  $\sigma$  changes by  $\Delta\sigma$ , the change in correntropy is less than  $\Delta\sigma/\sqrt{2\pi}\sigma_E$ . This weak sensitivity of the correntropy estimate w.r.t.  $\sigma$  is in contrast with the large dependence of the density estimation w.r.t.  $\sigma$ .

Furthermore, if the Gaussian assumption holds in regression problems, maximizing  $v_\sigma$  in Eq. (10.20) is essentially minimizing the error variance  $\sigma_E$ , and  $\sigma$  becomes immaterial provided the variance of the estimator (Eq. (10.19)) is reasonably upper-bounded. For instance, given  $N$  we can choose  $\sigma$  so that

$$\left| \sqrt{\text{var}(\hat{v}_{N,\sigma})}/\mathbb{E}(\hat{v}_{N,\sigma}) \right| < 0.05.$$

Asymptotically ( $N$  sufficiently large and  $\sigma$  small according to Eq. (10.19)) we have

$$\left| \sqrt{\text{var}(\hat{v}_{N,\sigma})}/\mathbb{E}(\hat{v}_{N,\sigma}) \right| \approx \sqrt{(N\sigma p_E(0))^{-1} \int G_1^2(u) du}. \quad (10.21)$$

**Property 10.7.** Correntropy is a “summary” second-order statistic of the data projected in a reproducing kernel Hilbert space (RKHS) defined by the kernel function  $\kappa$ .

*Proof.* According to Mercer’s theorem [217], any symmetric positive definite kernel function possesses an eigendecomposition as

$$\begin{aligned}\kappa(x, y) &= \sum_{n=0}^{\infty} \lambda_n \varphi_n(x) \varphi_n(y) = \langle \Phi(x), \Phi(y) \rangle \\ \Phi : x &\mapsto \sqrt{\lambda_n} \varphi_n(x), \quad n = 1, 2, \dots,\end{aligned}$$

where  $\{\varphi_n(x), n = 1, 2, \dots\}$  and  $\{\lambda_n, n = 1, 2, \dots\}$  are sequences of eigenfunctions and corresponding eigenvalues of  $\kappa(x, y)$ , respectively, and  $\langle \cdot, \cdot \rangle$  denotes the inner product between two infinite-dimensional vectors  $\Phi(x)$  and  $\Phi(y)$ . By the Moore–Aronszajn Theorem [7],  $\kappa(x, y)$  uniquely determines a high-dimensional reproducing kernel Hilbert space, denoted as  $H_\kappa$ , where the nonlinear transformation  $\Phi$  maps the original signals onto the surface of sphere in  $H_\kappa$ . Based on the symmetric positive definite kernel function  $\kappa(x, y)$ , the correntropy can also be interpreted as

$$v(X, Y) = E[\kappa(x, y)] = E[\langle \Phi(x), \Phi(y) \rangle_{H_\kappa}] = E[\Phi(x)^T \Phi(y)], \quad (10.22)$$

Assume the dimension of the feature space is  $M$  (eventually infinity as in the case of Gaussian kernel) and the kernel mapping is  $\Phi(X) = [\varphi_1(X) \varphi_2(X) \dots \varphi_M(X)]^T$ .

The second-order statistics between  $\Phi(X)$  and  $\Phi(Y)$  is expressed by the following correlation matrix

$$R_{XY} = E[\Phi(X)\Phi(Y)^T] = \begin{bmatrix} E[\varphi_1(X)\varphi_1(Y)] & \cdots & E[\varphi_1(X)\varphi_M(Y)] \\ \vdots & \ddots & \vdots \\ E[\varphi_M(X)\varphi_1(Y)] & \cdots & E[\varphi_M(X)\varphi_M(Y)] \end{bmatrix} \quad (10.23)$$

Therefore, from Eq. (10.22),

$$v(X, Y) = E[\Phi(X)^T \Phi(Y)] = \text{trace}(R_{XY}). \quad (10.24)$$

The trace of  $R_{XY}$  is equal to the sum of the eigenvalues, which clearly shows that correntropy summarizes the second-order statistics in the feature space. Combining this property with Property 10.4, we can infer that the trace of the Gram matrix measures the probability density of  $X = Y$  in the input space, for small kernels.

**Property 10.8.** If  $X$  and  $Y$  are statistically independent,

$$v(X, Y) = \langle E[\Phi(X)], E[\Phi(Y)] \rangle_{H_\kappa}, \quad (10.25)$$

where  $\langle \cdot, \cdot \rangle_{H_\kappa}$  denotes the inner product in RKHS defined by the kernel.

*Proof.* Using the notation in Property 10.7,

$$\begin{aligned} v(X, Y) &= \mathbb{E} \left[ \sum_{i=1}^M \lambda_i \varphi_i(X) \varphi_i(Y) \right] = \sum_{i=1}^M \lambda_i \mathbb{E}[\varphi_i(X)] \mathbb{E}[\varphi_i(Y)] \\ &= \langle \mathbb{E}[\Phi(X)], \mathbb{E}[\Phi(Y)] \rangle_{H_\kappa} \end{aligned} \quad (10.26)$$

by the independence assumption. This completes the proof.

This property can be called uncorrelatedness in feature space and is an easily computable necessary (but not sufficient) condition for independence between  $X$  and  $Y$ . Additionally, this property can be interpreted in terms of PDF. If  $X$  and  $Y$  are independent,

$$p_{X,Y}(x, y) = p_X(x)p_Y(y) \quad (10.27)$$

Using kernels to estimate these PDFs,

$$\hat{p}_{X,Y,\sigma}(x, y) = \frac{1}{N} \sum_{i=1}^N G_\sigma(x - x_i) G_\sigma(y - y_i) \quad (10.28)$$

$$\hat{p}_{X,\sigma}(x, y) = \frac{1}{N} \sum_{i=1}^N G_\sigma(x - x_i) \hat{p}_{Y,\sigma}(y, y_i) = \frac{1}{N} \sum_{i=1}^N G_\sigma(y - y_i). \quad (10.29)$$

Integrating Eq. (10.27) along the line  $x = y$  and using Eqs. (10.28), and (10.29) yields

$$\frac{1}{N} \sum_{i=1}^N G_{\sigma\sqrt{2}}(x_i - y_i) \approx \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N G_{\sigma\sqrt{2}}(x_i - y_j) \quad (10.30)$$

which is a sample estimate approximation of Eq. (10.25). The approximation in Eq. (10.30) is due to the Parzen estimate. When  $\sigma$  tends to zero and the product  $N\sigma$  to infinity, strict equality holds.

Using the analogy of potential fields in Chapter 2, the term on the right-hand side of Eq. (10.30) is an estimate for the cross-information potential (CIP), therefore for independent random variables the estimator of correntropy is an estimator for the CIP, which is a great computational savings (one sum versus two sums). From the viewpoint of kernel methods,  $p_X(\cdot) = \mathbb{E}[\Phi(X)]$ ,  $p_Y(\cdot) = \mathbb{E}[\Phi(Y)]$  are two points in the RKHS, and the CIP is the inner product between the vectors created by these two PDFs, as demonstrated in Chapter 9. But if we recall the discussion in Chapter 2, the CIP is also the argument of the logarithm in Renyi's quadratic cross-entropy between  $p_X(\cdot)$  and  $p_Y(\cdot)$ , so there is also an information-theoretic interpretation of correntropy when the variables are independent.

If two random variables are independent, the scalar  $J = v(X, Y) - \text{CIP}(X, Y) = 0$ , which means that correntropy yields a “weak” test for independence, in the sense that it is just a necessary condition for statistical independence.  $J$  is easily estimated from the data using the estimators

of correntropy in Eq. (10.3) and of the CIP in Eq. (7.5). Further analysis and applications of this property in independent component analysis (ICA) have been pursued [7]. Equation (10.30) bears resemblance to the constrained covariance proposed by Gretton et al in [123], which is a strong measure of independence. These authors constrained the covariance operator in a closed ball of a reproducing kernel Hilbert space and converted the measure into a matrix norm of Gram matrices. However, our measure starts directly from Parzen estimates of PDFs and is a more intuitive, although weaker, measure of independence.

**Property 10.9.** The estimator of correntropy, induces a metric in the sample space. Given two vectors  $X = (x_1, x_2, \dots, x_N)^T$  and  $Y = (y_1, y_2, \dots, y_N)^T$  in the sample space, the function  $CIM(X, Y) = (G_\sigma(0) - v(X, Y))^{1/2}$  defines a metric in the sample space and is named as the correntropy induced metric (CIM). This property was proved in Chapter 3 and is included here just for completeness.

**Property 10.10.** For any symmetric positive definite kernel (i.e., Mercer kernel)  $\kappa(x, y)$  defined on  $R \times R$ , the cross-correntropy  $v(X, Y)$  is a reproducing kernel.

*Proof.* Because  $\kappa(x, y)$  is symmetrical, it is obvious that  $v(X, Y)$  is also symmetrical. Now, because  $\kappa(x, y)$  is positive definite, for any set of  $N$  points  $\{x_1, \dots, x_N\}$  and any set of real numbers  $\{\alpha_1, \dots, \alpha_N\}$ , not all zero

$$\sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j \kappa(x_i, y_j) > 0. \quad (10.31)$$

It is also true that for any strictly positive function  $g(\cdot, \cdot)$  of two random variables  $x$  and  $y$ ,  $E[g(x, y)] > 0$ . Then

$$\begin{aligned} E \left[ \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j \kappa(x_i, y_j) \right] &> 0 \Rightarrow \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j E[\kappa(x_i, y_j)] \\ &= \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j v(X, Y) > 0 \end{aligned}$$

Thus,  $v(X, Y)$  is both symmetric and positive definite. Now, the Moore–Aronszajn theorem [7] proves that for every real symmetric positive definite function of two real variables, there exists a unique reproducing kernel Hilbert space, denoted as  $H_v$ , with the positive function as its reproducing kernel.

As we have seen in Eq. (10.10), correntropy for small kernel sizes approaches  $p(X = Y)$ , (i.e. the integral of the joint PDF along the line  $x = y$ ). This interpretation enables a generalization of correntropy we call *parametric correntropy*.

**Definition.** Parametric correntropy in Cartesian coordinates is defined as

$$v_{a,b}(X, Y) = \mathbb{E}[G_\sigma(aX + b - Y)] = \iint G_\sigma(ax + b - y)p(x, y)dxdy \quad (10.32)$$

for any real  $a$  and  $b$ , and  $a \neq 0$ . This definition allows integration of joint PDFs on lines that do not necessarily pass through the origin and have arbitrary slopes.

For the special case of lines passing through the origin ( $b = 0$ ), parametric correntropy can also be defined as

$$v_\theta(X, Y) = \iint G_\sigma(\sin(\theta)x - \cos(\theta)y)p(x, y)dxdy, \quad (10.33)$$

where  $\theta \in [0, \pi]$  is the radial angle in the joint space and it defines the integral of the joint PDF along any line that passes through the origin with slope  $\theta$ . Correntropy is obtained for the special case of  $\theta = 45$  degrees in Eq. (10.33).

**Property 10.11.** The parametric correntropy that passes through the origin in the limit of  $\sigma \rightarrow 0$  is a radial conditional PDF and can be used to calculate a radial marginal PDF in the joint space.

*Proof.* Let us operate a change of variables from Cartesian to polar coordinates  $(x, y) = re^{j\theta}$ . The joint density can be written as  $p(R, \Theta) = p(\Theta)p(R|\Theta)$ . By definition the radial conditional density function  $p(r|\Theta = \theta)$  is a cut of the joint PDF along the radial direction  $\Theta = \theta$ . Integrating the joint density along this radial direction yields  $\int p(\Theta)p(r|\Theta = \theta)dr = p(\theta)$ . If we take the limit of  $\sigma \rightarrow 0$  in Eq. (10.33), we see that we obtain exactly the same value, provided the integral exists.

The radial marginal density function by definition will quantify the change of probability density in the joint space as a function of the angle. Therefore, it is possible to estimate a radial marginal density function with correntropy by computing  $\hat{p}(\Theta = \theta)$  and scanning  $0 \leq \theta < \pi$ . The values  $v_\theta(X, Y)|_{\theta=\pi/2}$  and  $v_\theta(X, Y)|_{\theta=\pi}$  represent the weighted sum of (marginal) moments for  $X$  and  $Y$ , respectively. The variables  $X$  and  $Y$  are relatively similar if  $v_\theta(X, Y)|_{\theta=\pi/4}$  is noticeably larger than the maximum of  $v_\theta(X, Y)|_{\theta=\pi/2}$  and  $v_\theta(X, Y)|_{\theta=\pi}$ . When  $v_\theta(X, Y)|_{\theta=\pi/4}$  is larger than  $v_\theta(X, Y)|_{\theta=\pi/2}$  it means that  $X$  and  $Y$  are more similar than  $X$  is to an all-zero signal. In practical cases it is not possible to make the kernel size zero as discussed in Property 10.4 because of finite data, but an approximation to the radial conditional probability and the marginal density function is possible.

Thus far we have addressed similarity between two random variables, but it is possible to discuss similarity between *two random vectors* of arbitrary dimension with correntropy if similarity is interpreted as similarity amongst all its pairwise components. Correntropy can be generalized for more than two

variables using the interpretation of correntropy expressed in Property 10.4. We consider not only probability along lines in two dimensions (as  $X = Y$ ) but probability along lines in  $L$ -dimensional spaces by using the Parzen estimator for the joint multidimensional PDF. The extension outlined here only applies to the radially symmetric Gaussian kernel. To simplify the presentation, the formulation will be done for the difference variable  $\mathbf{e}$ . The notation  $e_j(i)$  means the  $i$ th data sample of the  $j$ th component of the  $L$ -dimensional random vector  $\mathbf{e} = [e_1, e_2, \dots, e_L]^T$ . Suppose there are  $N$  data samples; then  $\hat{p}_E(e_1, e_2, \dots, e_L) = 1/N \sum_{i=1}^N \prod_{j=1}^L G_\sigma(e_j - e_j(i))$  or

$$\hat{p}_E(e_1, e_2, \dots, e_L) = \frac{1}{N} \sum_{i=1}^N \frac{1}{(2\pi\sigma^2)^{L/2}} \exp \left( -\frac{1}{2\sigma^2} \left( \sum_{j=1}^L (e_j - e_j(i))^2 \right) \right) \quad (10.34)$$

This extension of correntropy consists in a line integral defined by the bisector of the positive hyperoctant in  $L$ -dimensional space.

**Definition.** Correntropy for an  $L$ -dimensional random vector  $\mathbf{e} = [e_1, e_2, \dots, e_L]^T$  is

$$v(\mathbf{e}) = \int \dots \int \prod_{j=2}^L G_\sigma(e_1 - e_j) p_E(e_1, e_2, \dots, e_L) de_1 \dots de_L. \quad (10.35)$$

Equation (10.35) can be written as a simple integral in which all the parameters of the joint PDF are equal; that is,  $v(\mathbf{e}) = \int p_E(e, e, \dots, e) de$ . Therefore it can be easily estimated by plugging the estimator given in Eq. (10.34) to obtain

$$\begin{aligned} \hat{v}(\mathbf{e}) &= \frac{1}{N} \sum_{i=1}^N \int \frac{1}{(2\pi\sigma^2)^{L/2}} \exp \left( -\frac{1}{2\sigma^2} \left( \sum_{j=1}^L (e - e_j(i))^2 \right) \right) de \\ &= \frac{1}{N} \sum_{i=1}^N \int \frac{1}{(2\pi\sigma^2)^{L/2}} \exp \left( -\frac{1}{2\sigma^2} \left( \sum_{j=1}^L (e^2 - 2ee_j(i) + e_j(i)^2) \right) \right) de. \end{aligned} \quad (10.36)$$

The argument of the exponential involves a sum of second order terms that can be written as a product of a perfect square of polynomial terms multiplied by a reminder term as

$$\hat{v}(e) = \frac{1}{N} \sum_{i=1}^N \int \frac{1}{(2\pi\sigma^2)^{L/2}} \exp \left( -\frac{L}{2\sigma^2} (e^2 - 2e\gamma_i + \gamma_i^2) \right) \exp \left( -\frac{L}{2\sigma^2} C_i^2 \right) de, \quad (10.37)$$

where  $\gamma_i = \sum_{j=1}^L e_j(i)/L$  and  $C_i = \sqrt{\sum_{j=1}^L (e_j(i))^2/L - (\sum_{j=1}^L e_j(i)/L)^2}$  are the polynomial factors. Now let  $\bar{\sigma} = \sigma/\sqrt{L}$  and write the expression as

$$\hat{v}(\mathbf{e}) = \frac{1}{N} \sum_{i=1}^N \int \frac{1}{L^{L/2}} \frac{1}{(2\pi\bar{\sigma}^2)^{(L-2)/2}} \frac{1}{(2\pi\bar{\sigma}^2)^{1/2}} \exp\left(-\frac{1}{2\bar{\sigma}^2}(e - \gamma_i)^2\right) \frac{1}{(2\pi\bar{\sigma}^2)^{1/2}} \exp\left(-\frac{1}{2\bar{\sigma}^2}C_i^2\right) de.$$

As the integral is a complete Gaussian it integrates to one and simplifies to

$$\hat{v}(\mathbf{e}) = \frac{1}{N} \sum_{i=1}^N \frac{1}{L^{L/2}} \frac{1}{(2\pi\bar{\sigma}^2)^{(L-2)/2}} G_{\sigma'}(C_i).$$

Finally, plugging the value of  $C_i$  from the factorization yields

$$\hat{v}(\mathbf{e}) = \frac{1}{N} \sum_{i=1}^N \frac{1}{L} \frac{1}{(2\pi\bar{\sigma}^2)^{(L-2)/2}} G_{\bar{\sigma}} \left( \sqrt{\frac{1}{L} \sum_{j=1}^L e_j^2(i) - \left( \frac{1}{L} \sum_{j=1}^L e_j(i) \right)^2} \right). \quad (10.38)$$

### 10.3 Centered Cross-Correntropy and Correntropy Coefficient

In the previous section we defined cross-correntropy, but this function is not zero mean even if the input data are centered due to the nonlinear transformation produced by the kernel. In statistics, the covariance represents the second-order statistics of centered data, so we now define a “generalized” cross-covariance function, called *centered cross-correntropy* as

$$\begin{aligned} u(X, Y) &= \mathbb{E}_{X,Y}[G_{\sigma}(x, y)] - \mathbb{E}_X \mathbb{E}_Y[G_{\sigma}(x, y)] \\ &= \iint G_{\sigma}(x, y) (p_{XY}(x, y) - p_X(x)p_Y(y)) dx dy \\ &= \mathbb{E}[\langle (\Phi(x) - \mathbb{E}[\Phi(x)]), (\Phi(y) - \mathbb{E}[\Phi(y)]) \rangle_{H_{\kappa}}], \end{aligned} \quad (10.39)$$

where  $H_{\kappa}$  is the RKHS defined by the kernel. In the input space, the correntropy is the joint expectation of  $\kappa(x, y)$ , whereas centered correntropy is the joint expectation minus the marginal expectation of  $\kappa(x, y)$ . The centered cross-correntropy might be also interpreted as the trace of a covariance matrix for the transformed random variables in the high-dimensional RKHS  $H_{\kappa}$ , or equivalently, the trace of the cross-correlation matrix for the zero-mean (centered) random functions  $\Phi(x) - \mathbb{E}[\Phi(x)]$ , and  $\Phi(y) - \mathbb{E}[\Phi(y)]$ .

The sample mean estimator of the centering term,  $\mathbb{E}_X \mathbb{E}_Y[G(x, y)]$  is also numerically equal to the kernel estimator of the cross-information potential defined in Chapter 2, which as we saw is the argument of the log of Renyi’s quadratic cross-entropy.

Just by looking at Eq. (10.39) and utilizing our experience with kernel estimators it is straightforward to obtain an estimator for center correntropy  $u(X, Y)$  as

$$\hat{u}_\sigma(X, Y) = \frac{1}{N} \sum_{i=1}^N G_\sigma(x_i - y_i) - \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N G_\sigma(x_i - y_j). \quad (10.40)$$

**Property 10.12.** For symmetric kernels, the centered cross-correntropy  $u(X, Y)$  is a symmetric nonnegative definite function defined in  $X \times X \rightarrow R$ .

*Proof.* The symmetry of  $u(X, Y)$  is easily seen because the kernel function used in the definition is symmetric. Given any positive integer  $N$ , any set of  $\{x_1, x_2, \dots, x_N\}$  in  $X$  and any not all zero real numbers  $\alpha_1, \alpha_2, \dots, \alpha_N$ , by definition we have

$$\begin{aligned} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j u(x_i, x_j) &= \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j E[\langle (\Phi(x_i) - E[\Phi(x_i)]), (\Phi(x_j) - E[\Phi(x_j)]) \rangle_{H_\kappa}] \\ &= E \left[ \left\| \sum_{i=1}^N \alpha_i (\Phi(x_i) - E[\Phi(x_i)]) \right\|^2 \right] \geq 0 \end{aligned} \quad (10.41)$$

This is relevant because by Aronzajn's theorem it defines a RKHS. The importance of the centered correntropy in this discussion is to help us define a single scalar that can quantify the similarity of the two random variables, extending the concept of the correlation coefficient so pervasive in statistics.

**Definition.** The correntropy coefficient is defined as

$$\eta = \frac{u(X, Y)}{\sqrt{u(X, X)u(Y, Y)}}, \quad (10.42)$$

where  $u(X, Y)$  is the centered cross-correntropy defined in Eq. (10.39) and  $u(X, X)$  and  $u(Y, Y)$  are, respectively, the centered autocorrentropy functions for nondegenerate random variables  $X$  and  $Y$ , defined as  $u(X, X) = E_X[G_\sigma(x, x)] - E_X E_X[G_\sigma(x, x)]$ . A random variable  $X$  is nondegenerate (its PDF is not a delta function), if  $u(X, X) > 0$  [261].

**Property 10.13.** For nondegenerate random variables, the absolute value of the correntropy coefficient is bounded by 1; that is,  $|\eta| < 1$ .

*Proof.* This property can be proven by showing that the centered cross-correntropy  $u(X, Y)$  satisfies  $|u(X, Y)| \leq \sqrt{u(X, X)u(Y, Y)}$  which is simply the Cauchy–Schwarz inequality in the RKHS  $H_\kappa$ . Let  $n = 2$  in Eq. (10.41); the expression reduces to

$$\alpha_1^2 u(X, X) + \alpha_2^2 u(Y, Y) \geq 2\alpha_1 \alpha_2 |u(X, Y)|. \quad (10.43)$$

For nondegenerate random variables, we can substitute

$$\alpha_1^2 = \frac{u(Y, Y)}{2\sqrt{u(X, X)u(Y, Y)}} \quad \text{and} \quad \alpha_2^2 = \frac{u(X, X)}{2\sqrt{u(X, X)u(Y, Y)}}$$

into Eq. (10.43) to obtain Eq. (10.42). Therefore  $|u(X, Y)| \leq \sqrt{u(X, X)u(Y, Y)}$  must hold because the left-hand side is zero and the right-hand side is non-negative. Hence we conclude the proof.

**Property 10.14.** If the two random variables  $X$  and  $Y$  are independent, the correntropy coefficient reduces to zero.

*Proof.* This property is trivial to prove if we substitute in Eq. (10.39) the equality  $E_{X,Y}[G_\sigma(x, y)] = E_X E_Y[G_\sigma(x, y)]$  proven in Property 10.8.

Therefore, unlike the conventional correlation coefficient, the correntropy coefficient distinguishes between independence and uncorrelatedness. In fact, the correlation coefficient yields a zero value for both independent and uncorrelated random variables, and the correntropy coefficient yields necessarily zero if the random variables are independent, but will likely produce a nonzero value (which depends on the kernel width used in the Gaussian kernel) for two uncorrelated but not independent random variables.

In practice, only a finite number of data points are available from the real-world experiment. So the need arises to estimate the correntropy coefficient. Substituting the definition of the centered cross-correntropy Eq. (10.39) into the correntropy coefficient Eq. (10.42) and approximating the ensemble average by the sample mean, the following estimate of correntropy coefficient directly from data is obtained,

$$\hat{\eta} = \frac{\frac{1}{N} \sum_{i=1}^N G_\sigma(x_i, y_i) - \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N G_\sigma(x_i, y_j)}{\sqrt{\kappa(0) - \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N G_\sigma(x_i, x_j)} \sqrt{\kappa(0) - \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N G_\sigma(y_i, y_j)}}, \quad (10.44)$$

where  $N$  is the total number of samples, and  $(1/N^2) \sum_{i=1}^N \sum_{j=1}^N G_\sigma(x_i, y_j)$  is the estimator of the cross-information potential between  $X$  and  $Y$ ,  $(1/N^2) \sum_{i=1}^N \sum_{j=1}^N G_\sigma(x_i, x_j)$  and  $(1/N^2) \sum_{i=1}^N \sum_{j=1}^N G_\sigma(y_i, y_j)$  are the IP estimators for  $X$  and  $Y$ , respectively. In practice the estimate of the correntropy coefficient depends on the kernel size, and a theoretical analysis of its effect has not been conducted. Note the close resemblance of the correntropy coefficient of Eq. (10.42) with the Cauchy–Schwarz divergence between PDFs introduced in Chapter 2 (apart from the logarithm). In addition to the centering of the PDFs using the IP, the computation is only done on the line  $x = y$ .

## 10.4 Parametric Cross-Correntropy and Measures of Dependence

An important observation in the definition of the centered correntropy Eq. (10.39) is that when two random variables are independent ( $p_{X,Y}(x,y) = p_X(x)p_Y(y)$ ) the quantity becomes zero but not vice versa. In order to make it a suitable dependence measure, we modify the definition of the centered correntropy so that the correntropy coefficient satisfies the condition of attaining zero if and only if the two random variables are independent.

**Definition.** Given two random variables  $X$  and  $Y$ , the parametric centered correntropy is defined as

$$u_{a,b}(X, Y) = E_{X,Y}[G_\sigma(aX + b - Y)] = \iint G_\sigma(ax + b - y)(p_{XY}(x, y) - p_X(x)p_Y(y))dxdy, \quad (10.45)$$

where  $p_{XY}(x, y)$  is the joint PDF of random variables,  $p_X(x)$  and  $p_Y(y)$  are the marginal PDFs, respectively, and  $a \neq 0$  and  $b$  are parameters in  $R$ .

Property 10.10 illustrates the meaning of parametric correntropy in the input space as radial conditionals, but in the context of dependence measures, the importance of parametric centered correntropy comes from the following property.

**Property 10.15.** For infinite support kernels, the parametric centered correntropy is zero for all  $a$  and  $b$  if and only if  $X$  and  $Y$  are independent random variables.

This property is proved in [261] and involves the use of Bochner's theorem. Obviously this is a very important property because it links independence between scalar random variables with the calculation of a function of two parameters, for which we have already presented sample estimators.

**Definition.** Given two random variables  $x$  and  $y$ , and two real numbers  $a, b$  with  $a \neq 0$  the parametric correntropy coefficient is defined as

$$\begin{aligned} \eta_{a,b}(X, Y) &= \eta(aX + b, Y) = \frac{u(aX + b, Y)}{\sqrt{u(aX + b, aX + b)u(Y, Y)}} \\ &= \frac{E_{X,Y}[G_\sigma(ax + b, y)] - E_X E_Y[G_\sigma(ax + b, y)]}{\sqrt{E_Y[G_\sigma(ax + b, ax + b)] - E_X E_X[G_\sigma(ax + b, ax + b)]}\sqrt{E_Y[G_\sigma(y, y)] - E_Y E_Y[G_\sigma(y, y)]}}. \end{aligned} \quad (10.46)$$

When samples from each one of the random variables are available, the parametric correntropy coefficient can be estimated from samples using kernels as

$$\hat{\eta}_{a,b} = \frac{\frac{1}{N} \sum_{i=1}^N G_\sigma(ax_i + b, y_i) - \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N G_\sigma(ax_i + b, y_j)}{\sqrt{G_\sigma(0) - \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N G_\sigma(ax_i + b, ax_j + b)}\sqrt{G_\sigma(0) - \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N G_\sigma(y_i, y_j)}}. \quad (10.47)$$

**Property 10.16.** The absolute value of the parametric correntropy coefficient is bounded by 1; that is,  $|\eta_{a,b}| \leq 1$ .

The proof of this property was already given in Property 10.13.

This development showed that the parametric correntropy coefficient has the very interesting property that it is zero for all  $a$  and  $b$  if and only if the random variables are independent, and is always bounded between  $\pm 1$ . From these results it is possible to define a novel dependence measure that obeys most of the Renyi' postulates [262].

## Measures of Dependence

The concept of statistical independence is very clearly defined. However, when two variables are not independent, then they are dependent. How to properly quantify the degree of dependence is a central problem in statistics that has received a lot of attention for over a century. The difficulty is that a constructive definition of statistical dependence does not exist. Therefore, measures of dependence proliferate in the literature. They are mostly based on the distance between the joint PDF of the data and the product of the marginal PDFs [212]. Some of the most commonly used measures of bivariate dependence include but are not limited to the correlation coefficient, Kendall's  $\tau$ , Spearmans'  $\rho_s$ , maximal correlation coefficient, monotone correlation coefficient, and others.

The concept of dependence is also closely related to the amount of information that one random variable contains with respect to another, because the more dependent the random variables are, the more mutual information they share. Therefore, several measures based on information theory have been proposed in the literature, such as mutual information that can also be interpreted as the Kullback–Leibler divergence between the joint probability density functions and the product of the marginal densities [65]. It turns out the Kullback–Leibler divergence is a special case of the  $\varphi$ -divergences discussed in Chapter 1 when a specific convex function  $\varphi$  is chosen [218]. Silvey's generalized coefficient [301] uses the Radon–Nikodym derivative of joint distribution of random variables with respect to the product of their marginal distributions. Other dependence measures based on information theory include the relative entropy measure proposed by [169] and others. However, Joe's relative entropy dependence measure [169], and almost all other entropies fail to be “metric” because they violate the triangular inequality. To overcome this problem Granger, et al [119] proposed a metric measure of dependence.

Recently there has been considerable work on using functions in reproducing kernel Hilbert space to quantify dependence. Bach and Jordan [12] introduced kernel dependence functionals in the context of independent component analysis. The kernel canonical correlation is a kernelization of canonical correlation analysis with regularization on the functionals. The kernel generalized variance is an extension of kernel canonical correlation by estimating the

spectral norm of the correlation operator between reproducing kernel Hilbert spaces in the entire spectrum [12]. Instead of using the correlation operator in RKHS, Gretton et al. [123] proposed the kernel constrained covariance (COCO) and kernel mutual information based on the covariance operator. The kernel constrained covariance estimates the spectral norm of the covariance operator between RKHS. It has been proven that the kernel constrained covariance is zero if and only if the original random variables are independent provided that the kernel is universal. These dependence measures based on kernel methods have enjoyed much success in independent component analysis, quantification of generalized synchronization between chaotic signals, and other machine learning applications.

Renyi took a functional approach to the problem and proposed seven axioms that any measure of dependence should fulfill [262]. Accordingly, Renyi's seven postulates for a proper measure of dependence  $Q(X, Y)$  are:

1. Well defined over  $X, Y$ .
2. Symmetric  $Q(X, Y) = Q(Y, X)$ .
3. Bounded  $0 \leq |Q(X, Y)| \leq 1$ .
4.  $Q(X, Y) = 0$  iff  $X$  and  $Y$  are independent.
5.  $Q(X, Y) = 1$  if  $Y = f(X)$ .
6. Invariant  $Q(X, Y) = Q(f(X), g(Y))$  for any functions  $f, g$ .
7.  $Q(X, Y) = \rho(X, Y)$  for  $X, Y \sim N(0, \sigma^2)$ .

Renyi showed that one measure satisfying all these postulates is  $Q(X, Y) = \sup_{f,g} |\text{corr}(f(x), g(x))|$  where  $f(x)$  and  $g(x)$  are Borel measurable and have finite positive variance. The search over all pairs of functions makes maximal correlation impractical. Based on the parametric correntropy coefficient developed in the previous section, we formulate a novel dependence measure for two random variables that obeys postulates 1,3,4,5.

**Definition.** Given two random variables  $x$  and  $y$ , the correntropy dependence measure (CDM) is defined as

$$\Gamma(X, Y) = \sup_{a,b} |\eta_{a,b}(X, Y)|, \quad (10.48)$$

The proof that  $\Gamma(X, Y)$  fulfills the four postulates can be found in [261], but can also be easily derived from the previous properties. The second Renyi postulate for a dependence measure is symmetry. By defining  $\Gamma_N(X, Y) = (\Gamma(X, Y) + \Gamma(Y, X))/2$  we can make CDM symmetric. By using very large kernels, CDM can also approximate postulate 7, therefore the only outstanding postulate is 6. One of the interesting properties of CDM is that it achieves the limits of the range when the random variables are linearly dependent. This property is also shared by the correlation coefficient, and can be used to test hypotheses about linear dependence in experimental science.

Unlike the maximal correlation proposed by Renyi, CDM searches a two dimensional parameter space (for scalar random variables) for a maximum instead of the entire function space. With the incomplete Cholesky

decomposition explained in Chapter 2, the computational complexity of CDM is  $O(N)$  for each  $(a, b)$  pair. However, one of the difficulties of CDM is the poor scaling up with the dimensionality of the problem. Currently we have used simple grid search methods, which are fine for low dimensional problems, but the search can become prohibitive in high dimensions. It may be possible to exploit some structure in the joint space to speed up the search, but this has not been investigated.

Compared to the kernel constrained covariance presented in [123], our measure is very simple to understand and does not require regularization techniques, although the kernel size still needs to be chosen from the data. We address this point in the case studies. The kernel constrained covariance operates on the high (possibly infinite) dimensional functional space, therefore regularization is mandatory for good results. One of the appeals of COCO is that it can quantify both linear and nonlinear dependence, and the solution involves simple matrix operators on the Gram matrix, independent of the dimensionality of the input space, but the algorithm is  $O(N^2)$ . The approach we took for the CDM is interesting because it poses the problem in the input space using a novel perspective, and is very intuitive and practical for small-dimensional problems. For instance, the values of  $a, b$  that maximize (minimize) CDM have meaning because they show the direction in the input space where the variables are most similar (dissimilar). More generally, a dependence profile as a function of  $(a, b)$  can be used for design of practical experiments.

## 10.5 Application: Matched Filtering

Matched filtering is known to be optimal for detection of data sent through a linear channel in additive white Gaussian noise (AWGN) conditions. However, there are many important cases in practice (i.e., nonlinear channels or linear channels affected by “fat tail” noise) where the matched filter performs poorly. Inspired by the new correntropy induced metric of Property 10.9 between two random vectors, we define a new decision statistic used for matched filtering which we call the correntropy matched filter (CMF) [249].

Let us assume a receiver and a linear channel with white additive noise. For simplicity the binary detection problem is considered where the received vector is denoted  $\mathbf{r}$ . There are two possible cases: (a) when the signal  $\mathbf{s}_0$  is present (hypothesis  $H_0$ ,  $\mathbf{r}_0 = \mathbf{n} + \mathbf{s}_0$ ) and (b) when the signal  $\mathbf{s}_1$  is present (hypothesis  $H_1$ ,  $\mathbf{r}_1 = \mathbf{n} + \mathbf{s}_1$ ). Now it is sufficient to check whether the received vector is “closer” to  $\mathbf{s}_1$  (validating  $H_1$ ) or to  $\mathbf{s}_0$  (validating  $H_0$ ) based on the CIM similarity measure, which implies that cross-correntropy is appropriate. We model the received signals and templates as random variables, with  $N$  realizations for each (the length of each template  $\mathbf{s}_p$ ,  $p = 0, 1$ ). Using the intuition given by Property 10.4, we can consider the joint space of both the received signal and the known templates  $(\mathbf{r}, \mathbf{s}_0)$  and  $(\mathbf{r}, \mathbf{s}_1)$  and use

cross-correntropy to find out which is larger. The simplest hypothesis test is based on the sign of the difference, which yields the correntropy matched filter statistic

$$L_C(\mathbf{r}) = \frac{1}{N} \sum_{i=1}^N G_\sigma(\mathbf{r}_i, \mathbf{s}_{1,i}) - \frac{1}{N} \sum_{i=1}^N G_\sigma(\mathbf{r}_i, \mathbf{s}_{0,i}) \quad (10.49)$$

The case when timing is unknown is also simple and requires finding the best lag to make the decision [249].

### Selection of Kernel Size

The kernel size is a free parameter in information-theoretic methods, so it has to be chosen by the user. For the particular case of the correntropy matched filter we can show the following interesting property.

**Property 10.17.** For the Gaussian kernel, the decision under  $L_C$  (Eq. (10.49)) reduces to a maximum likelihood decision in the input space for i.i.d. Gaussian noise.

*Proof.* Using the Taylor series expansion for a Gaussian kernel we get

$$G_\sigma(r_n, s_{p,i}) = 1 - \frac{(r_n - s_{p,i})^2}{2\sigma^2} + \frac{(r_n - s_{p,i})^4}{2(2\sigma^2)^2} - \dots \quad p = 0, 1 \quad i = 1, \dots, N$$

Note that because the order of the exponent increases by two with each term, the contribution of the higher-order terms becomes less significant compared to the lower-order terms as  $\sigma$  increases. Truncating at the second term yields  $G_\sigma(r_n, s_{p,i}) \approx 1 - (r_n - s_{p,i})^2/2\sigma^2$ . On the other hand, the log-likelihood solution for the detection is  $L_{ML}(r) = 1/N \sum_{i=1}^N r_i(s_{1i} - s_{0i})$ , therefore  $L_C \propto L_{ML}$ .

This can be interpreted as saying that the kernel size acts as a way of tuning the correntropy matched filter, where larger kernel sizes are appropriate for linear channels and Gaussian noise. For instance, in AWGN, Property (10.17) implies that the kernel size should be relatively larger than the dynamic range of the received signal so that the performance defaults to that of the linear matched filter, which is optimal for this case. In the experiments presented below, the kernel size is chosen heuristically so that the best performance is observed. In the cases where the correntropy matched filter is expected to bring benefits, such as for additive impulsive noise, the kernel size should be selected according to density estimation rules to represent the template well (i.e., Silverman's rule [300]). As illustrated later, choosing the appropriate kernel size for the detection problem is not very critical (unlike many other kernel estimation problems) because any value over a wide interval provides nearly optimal performance for a given noise environment.

## Matched Filtering Case Studies

Receiver operating characteristic (ROC) curves were used to compare the performance of the linear matched filter (MF), the matched filter based on mutual information (MI) explained in [94], and the proposed correntropy matched filter (CMF). ROC curves give the plot of the probability of detection  $P_D$  against the probability of false alarm  $P_{FA}$  for a range  $[0, \infty)$  of threshold values,  $\gamma$  (the highest threshold corresponds to the ROC value for  $P_f = 0$ ). The area under the ROC can be used as a means of measuring the overall performance of a detector. The ROC curves were plotted for various values of the signal-to-noise ratio (SNR), defined as the ratio of the total energy of the signal template to the noise variance. For symmetric  $\alpha$ -stable distributions (defined by the characteristic function  $\Psi_\alpha(u) = \exp(-(\sigma|u|)^\alpha)$ ), where the variance of the noise is not defined, SNR was estimated as the signal power to squared scale of noise ratio. A total of 10,000 MonteCarlo (MC) simulations were run for each of the following cases. The single template is a sinusoid signal with length 64 given by  $s_i = \sin(2\pi i/10)$ ,  $i = 1, 2, \dots, 64$ . Segments of length  $N$ , some containing the signal and others without the signal, were generated with a 50% probability of transmitting a signal. The following combinations of noise and channel types are simulated here.

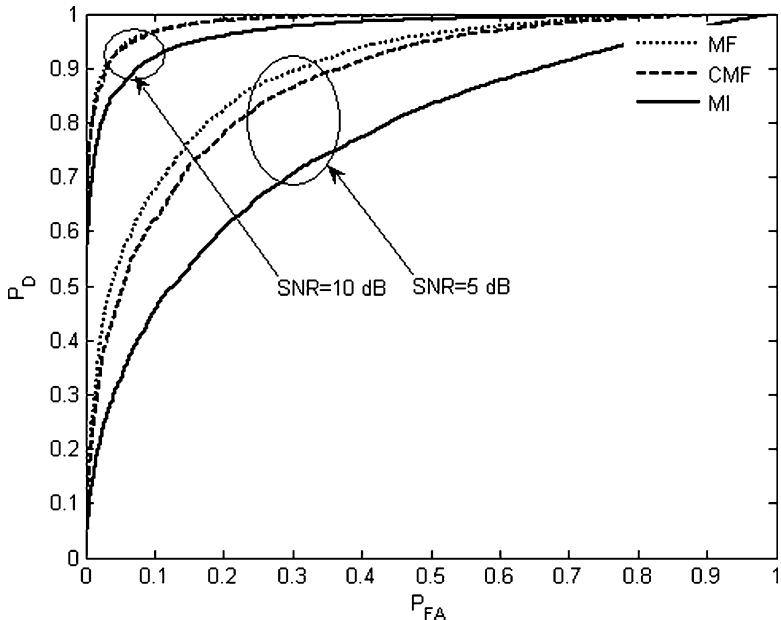
1. Additive white Gaussian noise linear channel.
2. Additive zero mean  $\alpha$ -stable noise channel.

$\alpha$ -stable distributions have been widely used to model heavy-tailed or leptokurtic distributions, especially those observed in financial data analysis [215], and are also a good noise model for underwater communications. Here we use the  $\alpha$ -stable distribution because of its gradual deviation from Gaussianity as  $\alpha$  decreases from 2 to 1 (when the distribution becomes Cauchy). This range is also appropriate because even though the higher moments diverge, the mean is still defined. The symmetric  $\alpha$ -distributed data are generated using the method given in [231] with the skewness parameter set to  $\text{beta} = 0$ .

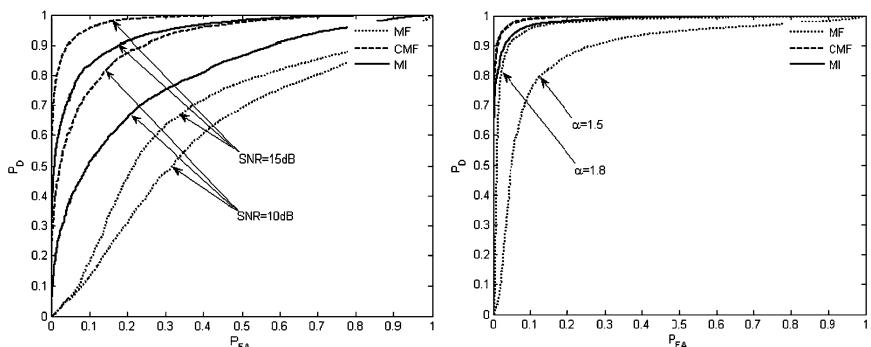
*AWGN and Linear Channels:* For the AWGN case, the matched filter is optimal so it should outperform all the other filters being tested. But the CMF, although obtained in a different way, provides almost the same performance as the matched filter as can be expected from Property 10.20, and observed in Figure 10.1. In fact, the CMF performance will approach the MF arbitrarily as the kernel size increases. The MI-based matched filter is not able to reach the performance of CMF even when the kernel size is increased to high values.

*$\alpha$ -Stable Noise in a Linear Channel:* Now let us observe the behavior of these methods for  $\alpha$ -stable distributed noise. Since these distributions have second moments tending to infinity for  $\alpha$  close to 1, the matched filter will utterly fail. Figure 10.2a shows the results for  $\alpha = 1.1$ .

Of course, as  $\alpha$  increases and approaches 2, the performance of the linear MF improves because the noise then becomes Gaussian (see Figure 10.2b).



**Fig. 10.1.** ROC plots for AWGN channel with kernel variance  $\sigma^2(CMF) = 15 \sigma^2(MI) = 15$  (the curves for MF and CMF for 10 dB overlap) (from [249]).



**Fig. 10.2. (a)** ROC plots for additive channel with white alpha-stable distributed noise, kernel variance  $\sigma^2 = 3$ ,  $\alpha = 1.1$ . **(b)** ROC plots for additive channel with several white  $\alpha$ -stable distributed noise, kernel variance  $\sigma^2 = 3$ , SNR = 15 dB; the plots for MI and CMF almost coincide for both values of  $\alpha$  (from [249]).

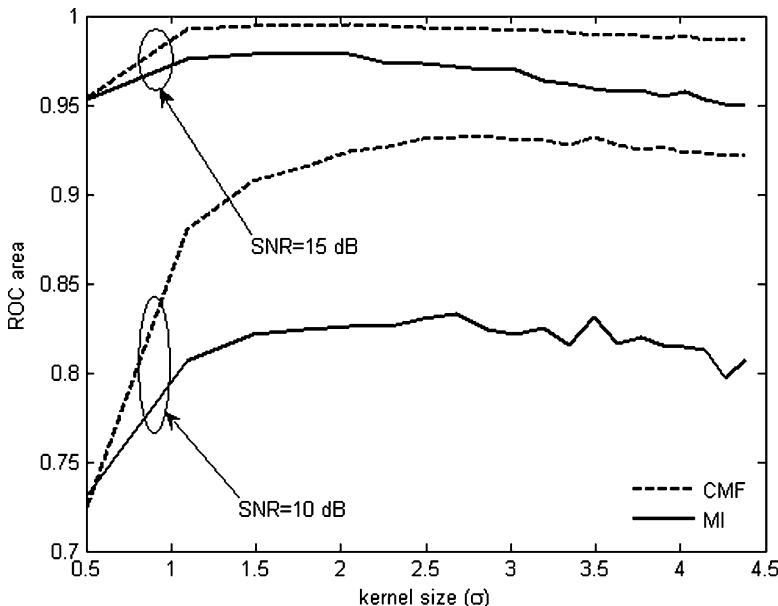
For this case because the variance of the noise is not defined, the SNR implies the squared scale of noise to signal power ratio.

### Effect of the Kernel Size

The choice of kernel size, although important, is not as critical as in many other kernel methods and density estimation problems. For detection with the correntropy matched filter, we plot the area under the curve for different values of the kernel size to evaluate its effect on detectability. As can be seen in Figure 10.3, a wide range of kernel sizes works well. Note that for the  $\alpha$ -stable noise, Silverman's rule is not computable because the variance of the noise is ill-defined. However, it is trivial to choose a value for kernel size using a quick scan on a training set to select the best performance for the particular application. Tests with impulsive noise (not shown) show that Silverman's rule finds a kernel size in the best performance range.

### Matched Filter Statistics in Feature Space

So far we introduced the correntropy matched filter from an information-theoretic (ITL) learning perspective. We can also derive Eq. (10.49) from kernel methods with a few assumptions. We transform the data from the input space to the kernel feature space and compute the likelihood ratio of



**Fig. 10.3.** The area under the ROC for various kernel size values for additive  $\alpha$ -stable distributed noise,  $\alpha = 1.1$  (from [249]).

the received signal for the case of i.i.d. zero-mean noise  $L(\mathbf{r}) = (\mathbf{r} - \mathbf{s}_1)^T (\mathbf{r} - \mathbf{s}_1) - (\mathbf{r} - \mathbf{s}_0)^T (\mathbf{r} - \mathbf{s}_0)$  in the feature space by using the kernel trick, but instead of using the original kernel  $\kappa$  we design a new kernel as the average of kernel evaluations,  $\bar{\kappa}$  defined by

$$\bar{\kappa}(\mathbf{x}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^N G_\sigma(x_i - y_i), \quad (10.50)$$

where  $\mathbf{x} = [x_1, x_2, \dots, x_N]^T$  and  $\mathbf{y} = [y_1, y_2, \dots, y_N]^T$  are the input vectors. Note that Eq. (10.50) is the cross-correntropy estimate and  $\bar{\kappa}$  is a valid kernel because it is a sum of symmetric and positive definite functions. In a sense we are designing a kernel (i.e., a RKHS and its similarity metric) tuned to the data we want to find. Hence  $\bar{\kappa}$  can be written as

$$\bar{\kappa}(\mathbf{x}, \mathbf{y}) = \bar{\Phi}^T(\mathbf{x}) \bar{\Phi}(\mathbf{y}), \quad (10.51)$$

where  $\bar{\Phi}$  maps the input vector to a possibly (depending on the kernel chosen) infinite-dimensional feature vector.

With the two possible transmitted vectors  $\mathbf{s}_0$  and  $\mathbf{s}_1$ , and accordingly  $\mathbf{r}_0$  and  $\mathbf{r}_1$  as the received signal in the input space, the corresponding received feature vectors will be  $\bar{\Phi}(\mathbf{r}_0)$ ,  $\bar{\Phi}(\mathbf{r}_1)$ , respectively. Now applying the likelihood statistic in the feature space yields

$$L_{1\bar{\Phi}}(\mathbf{r}) = (\bar{\Phi}(\mathbf{r}) - \bar{\Phi}_1)^T (\bar{\Phi}(\mathbf{r}) - \bar{\Phi}_1) - (\bar{\Phi}(\mathbf{r}) - \bar{\Phi}_0)^T (\bar{\Phi}(\mathbf{r}) - \bar{\Phi}_0), \quad (10.52)$$

where  $\bar{\Phi}_p = E\bar{\Phi}(\mathbf{r}_p)$ ,  $p = 0, 1$ . This expression assumes that the corresponding covariance matrices are equal and diagonal, which only happens if the noise component is additive and uncorrelated in the feature space. In general, this is just an assumption because of the coupling due to the nonlinear transformation, but it makes the result computationally tractable. Dropping the terms not depending on  $\bar{\Phi}(\mathbf{r})$  and rescaling Eq. (10.52) we get

$$L_{2\bar{\Phi}}(\mathbf{r}) = \bar{\Phi}_1^T \bar{\Phi}(\mathbf{r}) - \bar{\Phi}_0^T \bar{\Phi}(\mathbf{r}). \quad (10.53)$$

We still have to calculate the expected value in the feature space. If the vector length (template size)  $N$  is sufficiently large, we can use the following approximation.

$$\begin{aligned} \bar{\Phi}_p^T \bar{\Phi}(\mathbf{r}) &= \left\{ E[\bar{\Phi}_p(\mathbf{r}_p)] \right\}^T \bar{\Phi}(\mathbf{r}) = E_{\mathbf{r}_p} [\bar{\Phi}_p(\mathbf{r}_p)^T] \bar{\Phi}(\mathbf{r}) \\ &= E_{\mathbf{r}_p} \left[ \frac{1}{N} \sum_{i=1}^N \kappa(r_{pi}, r_i) \right] \approx E_{\mathbf{r}_p} E[\kappa(r_{pi}, r_i)] \\ &\approx E[\kappa(r_{pi}, r_i)] \approx \frac{1}{N} \sum_{i=1}^N \kappa(r_{pi}, r_i), \end{aligned} \quad (10.54)$$

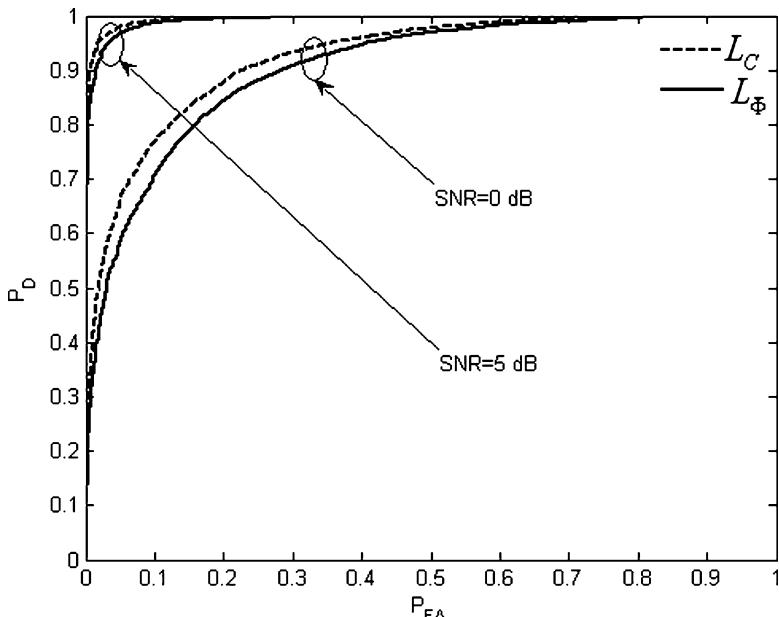
for  $p = 1, 0$ . So the modified decision statistic can be written as

$$L_{\bar{\Phi}} = \frac{1}{N} \sum_{i=1}^N \kappa(r_{1i}, r_i) - \frac{1}{N} \sum_{i=1}^N \kappa(r_{0i}, r_i), \quad (10.55)$$

where  $r_{1i} = s_{1i} + n_i$ ,  $r_{0i} = s_{0i} + n_i$ , and  $r_i = r_{1i}$  or  $r_{0i}$  depending on the signal that was sent. Equation (10.55) merely includes the lag zero estimate of cross-correntropy between  $\mathbf{r}$  and  $\mathbf{r}_1$  and that between  $\mathbf{r}$  and  $\mathbf{r}_0$ .

To compare the correlation receiver in feature space ( $L_{\bar{\Phi}}$ ) with the correntropy matched filter ( $L_C$ ), let the detection problem decide for simplicity if the signal  $\mathbf{s}$  was sent; that is,  $\mathbf{r}_1 = \mathbf{s} + \mathbf{n}$  and  $\mathbf{r}_0 = \mathbf{n}$ .

It can be seen that  $L_C$  is the same as  $L_{\bar{\Phi}}$  when the RKHS projected noise  $m$  instead of  $n$  corrupts the signal. Because the variance of  $m$  is greater than that of  $n$ ,  $L_C$  implements  $L_{\bar{\Phi}}$  for a greater SNR value. Thus using  $L_C$  results in improved performance. Note that this improvement is possible because the centers of the two hypotheses in RKHS don't solely consist of the template but also partly of the noise. This translates to using  $\mathbf{r}_1$  and  $\mathbf{r}_0$ , which are noisy versions of  $\mathbf{s}_0 = 0$  and  $\mathbf{s}_1 = \mathbf{s}$  in  $L_{\bar{\Phi}}$ , whereas  $L_C$  simply uses the "clean"  $\mathbf{s}_0 = 0$  and  $\mathbf{s}_1 = \mathbf{s}$  instead. This difference in performance is illustrated in Figure 10.4, which was generated using 10,000 Monte Carlo runs and shows the ROCs using  $L_{\bar{\Phi}}$  and  $L_C$  for impulsive noise for two different SNR values.



**Fig. 10.4.** ROC plots using  $L_{\bar{\Phi}}$  and  $L_C$  as the decision statistics using a channel with additive white impulsive noise for two SNR values, kernel variance  $\sigma^2 = 5$  (from [249]).

It should also be noted that the implementation of the cross-correntropy statistics using  $L_{\Phi}$  requires storage of a set of noise samples to compute  $\mathbf{r}_0$  and  $\mathbf{r}_1$  which is not needed for  $L_C$ . Interestingly,  $L_{\Phi}$  does not result in the maximum likelihood statistics in the feature space, because the data in the feature space are not guaranteed to be Gaussian. The correntropy matched filter happens to be a powerful and practical system for nonlinear channels or non Gaussian noise environments. Although we compared it with the simplest of the methods to ensure that the computational complexity is similar, the CMF can compete with more advanced designs.

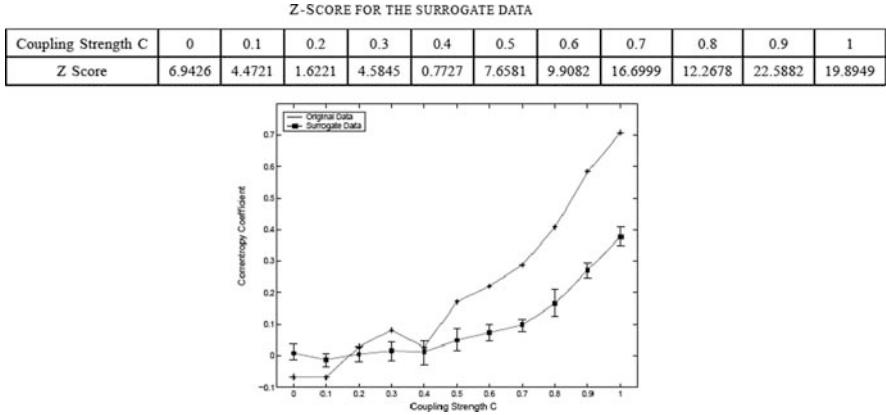
## 10.6 Application: Nonlinear Coupling Tests

The goal in this section is to demonstrate that the correntropy coefficient of Eq. (10.42) is able to statistically quantify nonlinear couplings between dynamical systems. In [344] we present a more in-depth study that shows that the correntropy coefficient is sensitive to abrupt changes of the coupling, and it is also able to estimate the strength of nonlinear coupling. Two Henon maps  $X$  and  $Y$  were unidirectionally coupled as

$$\begin{aligned} X : & \begin{cases} x_{n+1} = 1.4 - x_n^2 + b_x u_n \\ u_{n+1} = x_n \end{cases} \\ Y : & \begin{cases} y_{n+1} = 1.4 - [Cx_n + (1 - C)y_n] y_n + b_y v_n \\ v_{n+1} = y_n. \end{cases} \end{aligned}$$

Notice that system  $X$  drives system  $Y$  with a nonlinear coupling strength  $C$  where  $C$  ranges from 0 (no coupling) to 1 (complete coupling). Parameters  $b_x$  and  $b_y$  are both set to 0.3, the canonical value for the Henon map simulations. For each coupling strength, we discard the first 1000 outputs in as much as they are transient, and retain the next 500 data points for the experiments. The correntropy coefficient estimator of Eq. (10.43) is calculated between the first component  $x_1$  of system  $X$ , and the first component  $y_1$  of system  $Y$ .

In order to demonstrate that the correntropy coefficient is able to quantify the nonlinear coupling, we use the multivariate surrogate data technique introduced in [250]. Basically, in order to generate the multivariate surrogate data, first the Fourier transform is applied to each time series, then a random number is added to each of the phases, and an inverse Fourier transform is applied. The resulting time series have identical power spectra and cross-power spectra as the original time series, but any nonlinear coupling among the time series has been destroyed. In the simulation, we use the TISEAN package [291] to generate 19 realizations of the surrogate time series  $x_n$  and  $y_n$  for each different coupling strength. Then we compute the correntropy coefficient for both the original and the surrogate data with respect to different coupling strength. Figure 10.5 plots the correntropy coefficient curve for the original



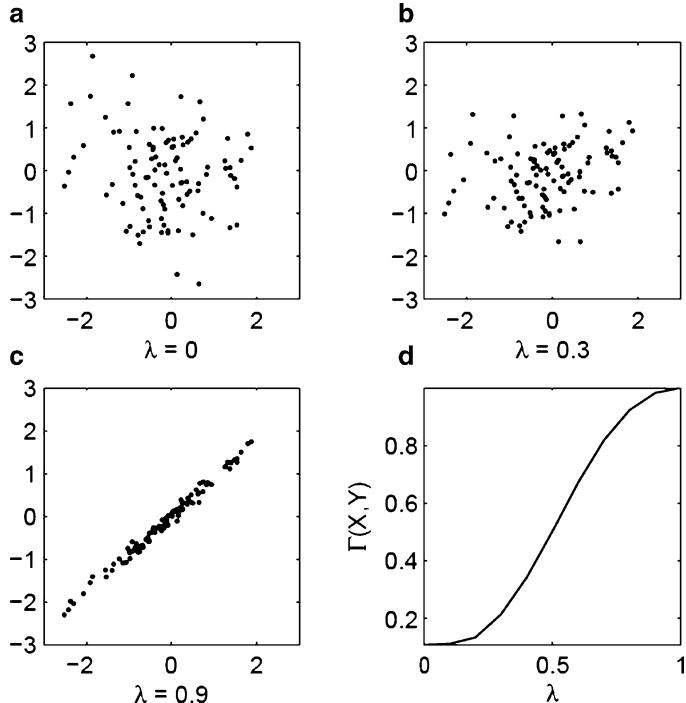
**Fig. 10.5.** Comparison of the correntropy coefficient for the original data and the surrogate data for unidirectionally coupled nonidentical Henon map (from [344]).

data and the mean value of 19 correntropy coefficients for the surrogate data with the corresponding maximal and minimal values indicated as error bars.

To quantify the significance level, we calculate the Z-score as  $Z = |v_{\text{orig}} - \mu_{\text{surr}}| / \sigma_{\text{surr}}$  where  $v_{\text{orig}}$  is the correntropy coefficient value for the original data and  $\mu_{\text{surr}}$  and  $\sigma_{\text{surr}}$  are the mean and the standard deviation for the surrogate data, respectively. The table in Figure 10.5 presents the Z-score values for different coupling strength. With the exception of  $C = 0.2$  and  $0.4$ , the Z-score values are significantly larger than 1.96 which means the nonlinear coupling has been detected with a probability  $p < 0.05$ . These results clearly demonstrate that the correntropy coefficient is sensitive to the nonlinearity of the dependence between two coupled systems. Note, however, that there are values of the correntropy coefficient that are higher and lower than the corresponding values of the surrogates, which complicates the assessment. We also experimentally addressed how the kernel size affects performance. We found that the statistical difference between the correntropy of the surrogates and the original system dynamics is only weakly coupled with the kernel size, which implies that the bias introduced by the kernel is approximately the same in the surrogates and in the original data.

## 10.7 Application: Statistical Dependence Tests

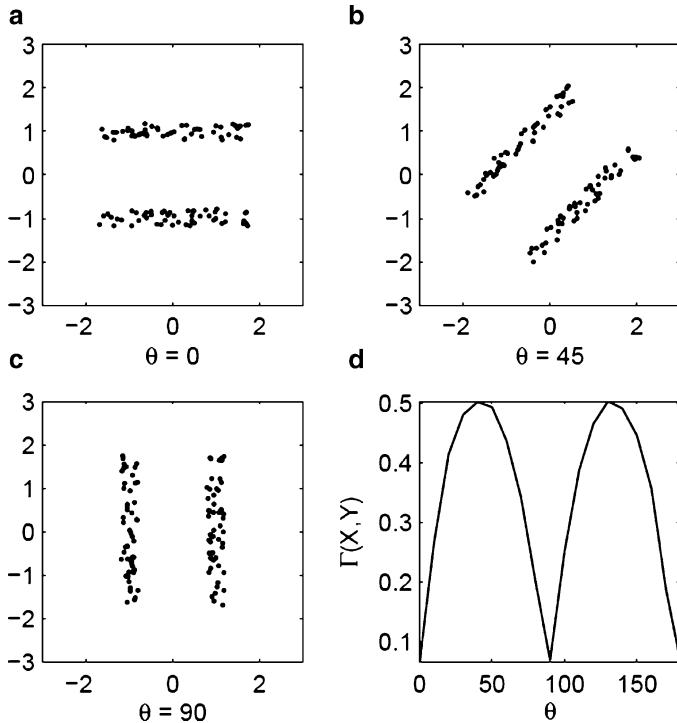
To demonstrate the validity of the proposed dependence measure, we start by applying it to a very basic problem. Consider two independent Gaussian random variables  $X$  and  $Y$  with zero mean and unit variance and construct a new random variable  $Z = \lambda X + (1-\lambda)Y$  with  $0 \leq \lambda \leq 1$ . When  $\lambda = 0$ ,  $Z = Y$ , and so  $Z$  and  $X$  are independent. However, when  $\lambda = 1$ ,  $Z = X$  and, therefore,



**Fig. 10.6.** The figures show the effect of  $\lambda$  on the dependence between  $X$  and  $Z$ . (a) The original random variables with  $\lambda = 0$ ; (b) with  $\lambda = 0.3$ ; and (c) with  $\lambda = 0.9$ ; (d) shows the change in the (measured) dependence value as a function of  $\lambda$  (from [261]).

$Z$  and  $X$  are the same random variable and, thus, they are strictly dependent. As we keep changing  $\lambda$  slowly from 0 to 1 we should expect a gradual increase in the dependence between  $Z$  and  $X$ . Figure 10.6a, b, and c show the samples from the scatterplot for random variable pair  $X$  and  $Z$  for different values of  $\lambda$ . Clearly, when  $\lambda = 0$  the random variables are independent whereas when  $\lambda = 0.3$  and  $\lambda = 0.9$ , they are more and more dependent. Figure 10.6d shows the variation of CDM as a function of  $\lambda$ . As  $\lambda$  is increased from 0 to 1, we notice a monotonic increase in the dependence measure  $\Gamma(X, Z)$ . We show in [261] that the kernel size does not affect the range of CDM, only the slope of the curve.

Next, we apply the CDM on a slightly more challenging problem, discussed in [107]. Consider two independent random variables  $X$  and  $Y$ , both having zero mean and unit variance, where  $X$  is uniformly distributed and  $Y$  is a combination of two uniform random variables each having equal probability of occurrence but not sharing a common support. (see Figure 10.7a.) This pair of random variables has an identity covariance matrix. Therefore if we generate a new pair of random variables by rotating this random variable pair by  $\theta$ ,



**Fig. 10.7.** The figures show how rotating the samples from two independent random variables affects the dependence between them. (a) The original random variables with zero rotation, (b) with  $45^\circ$  rotation, and (c) with  $90^\circ$  rotation. (d) shows the change in the (measured) dependence value as a function of the rotation (from [261]).

the covariance matrix does not change and, thus, the correlation between the new pair of random variables remains constant at zero. However, the dependence between the random variables changes with the rotation. Figures 10.7b and c show the samples from the random variable pair  $X$  and  $Y$  after rotation. Clearly, when  $\theta = 0^\circ$  or  $\theta = 90^\circ$  the new pair of random variables is independent whereas when  $\theta = 45^\circ$  the pair is highly dependent. Figure 10.7d reflects this fact. As we vary  $\theta$  we can notice the smooth variation in the dependence value between the resulting pair of random variables.

## 10.8 Conclusions

This chapter introduced a new function called cross-correntropy and its centered version as well as the cross-correntropy coefficient. These functions define generalizations for conventional correlation, covariance functions, and the correlation coefficient which are fundamental in statistics, pattern recognition, and machine learning. Therefore the impact of the newly defined functions

deserves a full investigation. We started the chapter by providing the most important properties that are currently known about these functions to set the foundations for the applications. Perhaps the most interesting feature of correntropy is that it provides a different measure of similarity that can be traced to the combination of higher-order statistics of the data that are expressed by the kernel. In the input space the idea is also clear. Similarity is evaluated along the bisector of the first and third quadrants, in a region defined by the kernel bandwidth, effectively being an estimation of  $p(X = Y)$ . This implies that the correntropy coefficient has the nice property of distinguishing between uncorrelated and independent variables, which is impossible to do with the correlation coefficient.

From the extension of these definitions to their parametric versions, we are able to address a central problem in statistics which is the quantification of statistical dependence. In fact the parametric correntropy coefficient (defined from the parametric centered crosscorrentropy) can be used to define the correntropy dependence measure (CDM) which has the appealing property that it is zero if and only if the pair of random variables is independent, and it becomes 1 or  $-1$  if the random variables are linearly dependent. This is achieved by searching over the space of the parameters (two for scalar r.v.), and for low-dimensional problems is still manageable and simple to understand.

Our case studies give just a glimpse of the possible applications of correntropy in statistical inference. We start by applying correntropy to improve the performance of matched filters for non-Gaussian and nonlinear channels and show the performance improves drastically for fat-tail or impulsive noise. The cross-correntropy coefficient is much more powerful than the correlation coefficient for quantifying nonlinear coupling as demonstrated in the example with the coupled Henon maps because of its sensitivity to the higher-order statistics of the time series. Finally, we test the results of CDM to measure dependence and also as a test for independence.

# Correntropy for Random Processes: Properties and Applications in Signal Processing

Puskal Pokharel, Ignacio Santamaría, Jianwu Xu, Kyu-hwa Jeong, and Weifeng Liu

## 11.1 Introduction

The previous chapter defined cross-correntropy for the case of a pair of scalar random variables, and presented applications in statistical inference. This chapter extends the definition of correntropy for the case of random (or stochastic) processes, which are index sets of random variables. In statistical signal processing the index set is time; we are interested in random variables that are a function of time and the goal is to quantify their statistical dependencies (although the index set can also be defined over inputs or channels of multivariate random variables). The autocorrelation function, which measures the statistical dependency between random variables at two different times, is conventionally utilized for this goal. Hence, we generalize the definition of autocorrelation to an autocorrentropy function. The name *correntropy* was coined to reflect the fact that the function “looks like” correlation but the sum over the lags (or over dimensions of the multivariate random variable) is the information potential (i.e., the argument of Renyi’s quadratic entropy). The definition of cross-correntropy for random variables carries over to time series with a minor but important change in the domain of the variables that now are an index set of lags. When it is clear from the context, we simplify the terminology and refer to the different functions (autocorrentropy, or cross-correntropy) simply as *correntropy function*, but keep the word “function” to distinguish them from Chapter 10 quantities.

In principle, the autocorrentropy function can substitute the autocorrelation function, but inasmuch as it is nonlinearly related to the data, it is important to analyze its properties to help us understand its benefits in nonlinear, non-Gaussian stationary signal analysis. As we show, the autocorrentropy function defines a new RKHS which is potentially useful because it can be used to implement nonlinear algorithms in the input space while solving linear algorithms in the RKHS. However, this RKHS is sufficiently different from the one induced by kernel methods that it requires a step-by-step explanation of how optimal algorithms can be mapped onto it, and how practical

optimal solutions are achieved (under some conditions). We show applications in speech processing to estimate the pitch, perform detection of sinewaves in impulsive noise regimes, implement an ICA algorithm based on the correntropy function, and design an object recognition algorithm.

## Correntropy in Time Series Analysis

The fact that reproducing kernels are covariance functions as pointed out by Aronszajn [8] and Parzen [238] explains their early role in time series analysis. However, the most recent work in kernel learning including support vector machines [323], kernel principal component analysis [288], kernel Fisher discriminant analysis [219], and kernel canonical correlation analysis [134] are based on similarity between identically independent distributed (i.i.d.) data, which in many cases is not realistic.

A large class of problems in engineering deal with time series having a complicated structure, which are normally modeled as *random processes*. By definition, a random process is a collection of random variables, that is,  $\{X_t, t \in T\}$  with  $T$  being an index set (integers for digital, or real values for continuous signal processing) and  $x_t \in R^d$ . Random processes are normally characterized by the statistical distribution of the instantaneous random variables' amplitudes and, unlike random variables, by their time structure. For this reason researchers developed measures to quantify the statistical time structure such as the autocorrelation function which is defined as  $R(t_1, t_2) = \text{Ex}[x_{t_1}x_{t_2}], \{t_1, t_2 \in T\}$ .

A major simplification occurs when the random process is wide-sense stationary, that is, when its first-order moment is independent of time,  $\text{E}[x_t] = \text{constant}, \{t \in T\}$ , and the autocorrelation function (second-order moment) is dependent only on the difference between the two selected time instances, that is,  $\text{E}[x_{t_1}x_{t_2}] = \text{E}[x_0x_{t_1-t_2}], \{t_1, t_2 \in T\}$ , which can be also expressed as  $R(t_1, t_2) = R(\tau)$  for  $\tau = t_1 - t_2$ . A single statistical measure that expresses relevant information from both the instantaneous and temporal structure of signals and that goes beyond second-order statistics would greatly enhance the theory of stochastic random processes.

The autocorrentropy function is a step forward in this direction. Specifically, it defines a generalized correlation function (GCF) in terms of inner products of vectors in a kernel feature space. Because inner products are a measure of similarity, for each two time indices in the input space defining a time lag, the autocorrentropy function in effect measures the pairwise similarity of the feature vectors. The inclusion of the time lag as a parameter in the GCF represents a fundamental change from the cross-correntropy. The former is now a function of the time lag, whereas the latter measures statistical information of a scalar random variable or vector at a single instant of time. From an analysis point of view, the autocorrentropy function maps the mean variability over the realizations calculated between two preselected time

intervals (lags) into a point in feature space. When the procedure is replicated for many lags, the effective dimension of the feature space is increased and it quantifies progressively better long-range statistical dependencies in the random process.

## 11.2 Autocorrentropy Function: Definition and Properties

**Definition.** Let  $\{X_t, t \in T\}$  be a strictly stationary stochastic process (i.e., the joint PDF  $p(x_{t_1}, x_{t_2})$  is unaffected by a change of the time origin) with  $T$  being an index set and  $x_t \in R^d$ . The autocorrentropy function  $v(t_1, t_2)$  is defined as a function from  $T \times T \rightarrow R^+$  given by

$$v_{x,x}(t_1, t_2) = E[\kappa(\mathbf{x}_{t_1}, \mathbf{x}_{t_2})] = \int \cdots \int \kappa(\mathbf{x}_{t_1}, \mathbf{x}_{t_2}) p(\mathbf{x}_{t_1}, \mathbf{x}_{t_2}) d\mathbf{x}_{t_1} d\mathbf{x}_{t_2}, \quad (11.1)$$

where  $E[\cdot]$  denotes mathematical expectation over the stochastic process  $\mathbf{x}_t$ , and  $\kappa$  is any continuous positive definite kernel.

Notice the similarity of Eq. (11.1) with that of cross-correntropy (Eq. (10.1)), except that now we are working with an index set of random variables at two different time instances of the same random process. Therefore this correntropy is truly an autocorrentropy function in time. Again, for kernels in  $L_\infty$  the autocorrentropy function always exists. When there is no source of confusion, the dependence on  $x$  is dropped from the notation. We pursue the analysis with symmetric translation invariant kernels such as the Gaussian kernel and scalar random variables so that Eq. (11.1) becomes

$$v_{x,x;\sigma}(t_1, t_2) = E[G_\sigma(x_{t_1}, x_{t_2})] = \int \int G_\sigma(x_{t_1} - x_{t_2}) p(x_{t_1}, x_{t_2}) dx_{t_1} dx_{t_2}. \quad (11.2)$$

Performing a Taylor series expansion (subject to the same restrictions of Eq. (11.1)), the autocorrentropy function of Eq. (11.2) can be rewritten as

$$v_\sigma(t_1, t_2) = \frac{1}{\sqrt{2\pi}\sigma} \sum_{n=0}^{\infty} \frac{(-1)^n}{2^n \sigma^{2n} n!} E \left[ \|x_{t_1} - x_{t_2}\|^{2n} \right] \quad (11.3)$$

which involves all the even-order moments of the random variable  $(x_{t_1} - x_{t_2})$ . Specifically, the term corresponding to  $n = 1$  in Eq. (11.3) is proportional to

$$E[||x_{t_1}||^2] + E[||x_{t_2}||^2] - 2E[\langle x_{t_1}, x_{t_2} \rangle] = \text{var}_{x_{t_1}} + \text{var}_{x_{t_2}} - 2R_x(t_1, t_2). \quad (11.4)$$

where  $R_x(t_1, t_2)$  is the covariance function of the random process; this shows that the information provided by the conventional covariance function (or the autocorrelation for zero-mean processes) is included within the

autocorrentropy function. For large kernel sizes the second-order moments dominate the autocorrentropy function and it defaults to biased autocorrelation, just as we saw for cross-correntropy in Chapter 10.

From Eq. (11.3), we can also see how the autocorrentropy function depends upon the higher-order moments of the random variables produced by the kernel (i.e., for the Gaussian kernel all even moments are created, whereas the Laplacian kernel creates all moments), therefore in order to have an univariate autocorrentropy function, all the moments must be invariant to time shifts. This is a more restrictive condition than the commonly used wide-sense stationarity (w.s.s.), which involves only second-order moments. More precisely, a sufficient condition to have  $v(t, t - \tau) = v(\tau)$  is that the input stochastic process must be strictly stationary on all moments [235]; this means that the joint PDF  $p(x_t, x_{t+\tau})$ , must be unaffected by a change of time origin. We assume this condition in the rest of the chapter when using  $v(\tau)$ .

For a discrete-time strictly stationary stochastic process we define the autocorrentropy function as  $v(m) = E[G_\sigma(x_n - x_{n-m})]$ , which can be easily estimated with kernels through the sample mean as

$$\hat{v}(m) = \frac{1}{N - m + 1} \sum_{n=m}^N G_\sigma(x_n - x_{n-m}). \quad (11.5)$$

The autocorrentropy function properties are similar to the cross-correntropy properties with the proper modification due to the differences in definitions: recall that cross-correntropy is defined on a pair of random variables, whereas the autocorrentropy function is defined on a pair of indexed sets, possibly continuous and of infinite dimension; the joint probability density function in cross-correntropy is over two random variables (or random vectors) at a single point in time, whereas the joint space in the autocorrentropy function is the space created by random variables at different lags. Therefore, crosscorrentropy computes a number, whereas the autocorrentropy function produces a number for each lag (i.e. it is a function of the lag).

The following properties are derived for Eq. (11.2) assuming a strictly stationary discrete-time stochastic process, but obviously, the properties are also satisfied for continuous-time processes.

**Property 11.1.** For symmetric kernels,  $v(m)$  is a symmetric function:  $v(m) = v(-m)$ .

**Property 11.2.**  $v(m)$  reaches its maximum at the origin (i.e.,  $v(m) \leq v(0) \quad \forall m$ ).

**Property 11.3.**  $v(m) \geq 0$  and  $v(0) = 1/\sqrt{2\pi}\sigma$ .

All these properties can be easily proven. Properties 11.1 and 11.2 are also satisfied by the conventional autocorrelation function, whereas Property 11.3 is a direct consequence of the positiveness of the Gaussian kernel.

**Property 11.4.** Given  $v(m)$  for  $m = 0, \dots, P-1$ , then the following Toeplitz matrix of dimensions  $P \times P$

$$\mathbf{V} = \begin{bmatrix} v(0) & \cdots & v(P-1) \\ \vdots & \ddots & \vdots \\ v(P-1) & \cdots & v(0) \end{bmatrix} \quad (11.6)$$

called the autocorrentropy matrix, is positive definite.

*Proof.* Using the empirical mean, the matrix  $\mathbf{V}$  can be approximated as  $\hat{\mathbf{V}} = \sum_{n=m}^N \mathbf{A}_n$  where  $\mathbf{A}_n$  is given by

$$\mathbf{A}_n = \begin{bmatrix} G_\sigma(x_n - x_n) & \cdots & G_\sigma(x_n - x_{n-P-1}) \\ \vdots & \ddots & \vdots \\ G_\sigma(x_n - x_{n-P-1}) & \cdots & G_\sigma(x_n - x_n) \end{bmatrix}.$$

Since the Gaussian is a kernel that satisfies Mercer's conditions,  $\mathbf{A}_n$  is a positive definite matrix. Furthermore, the sum of positive definite matrices is also positive definite [235], hence  $\hat{\mathbf{V}}$  is positive definite (for all  $N$ ); Taking the limit of  $N \rightarrow \infty$ ,  $\hat{\mathbf{V}}$  will approach  $\mathbf{V}$ , which concludes the proof.

We would like to point out that these four important properties open the possibility of applying the autocorrentropy function to all signal processing methods that use conventional correlation matrices, which include signal and noise subspace decompositions, projections, spectral estimation, and the like.

**Property 11.5.** Let  $\{x_n \in R, n \in T\}$  be a discrete-time strictly stationary zero-mean Gaussian process with autocorrelation function  $r(m) = E[x_n x_{n-m}]$ . The autocorrentropy function for this process is given by

$$v(m) = \begin{cases} \frac{1}{\sqrt{2\pi}\sigma} & m = 0 \\ \frac{1}{\sqrt{2\pi(\sigma^2 + \sigma^2(m))}} & m \neq 0, \end{cases} \quad (11.7)$$

where  $\sigma$  is the kernel size and  $\sigma^2(m) = 2(r(0) - r(m))$ .

*Proof.* The autocorrentropy function is defined as  $v(m) = E[G_\sigma(x_n - x_{n-m})]$ . Because  $x_n$  is a zero-mean Gaussian random process, for  $m \neq 0$   $\Delta x_m = x_n - x_{n-m}$  is also a zero-mean Gaussian random variable with variance  $\sigma^2(m) = 2(r(0) - r(m))$ . Therefore

$$v(m) = \int_{-\infty}^{\infty} G_\sigma(\Delta x_m) \frac{1}{\sqrt{2\pi\sigma^2(m)}} \exp\left(-\frac{\Delta x_m^2}{2\sigma^2(m)}\right) d\Delta x_m. \quad (11.8)$$

Because we are using a Gaussian kernel with variance  $\sigma^2$ , Eq. (11.8) is the convolution of two zero-mean Gaussians of variances  $\sigma^2$  and  $\sigma(m)^2$ , which produces a Gaussian evaluated at the origin. This yields Eq. (11.7) immediately. Property 11.5 clearly shows that the autocorrentropy function conveys information about the lag structure of the stochastic process and also about the weighted sum of even moments via quadratic Renyi's entropy. As a consequence of Property 11.5, if  $\{x_n \in R, n \in T\}$  is a white zero-mean Gaussian process strictly stationary with variance  $\sigma_x^2$  we have that  $v(m) = 1/\sqrt{2\pi(\sigma^2 + \sigma_x^2)}, m \neq 0$ , which is the mean value of the autocorrentropy function and is also the information potential of a Gaussian random variable of variance  $\sigma_x^2$ , when  $\sigma = \sigma_x^2$ .

The cross-correntropy of pairs of random variables defined a RKHS  $H_v$ , but its application is restricted because the two random variables are mapped to two functions in  $H_v$  so the effective dimension of the RKHS is 2. Using the autocorrentropy function a much more useful RKHS can be defined.

**Property 11.6.** For any symmetric positive definite kernel (i.e., Mercer kernel)  $\kappa(x_{t1}, x_{t2})$  defined on the index set  $T \times T$ , the autocorrentropy function defined in Eq. (11.2) is a reproducing kernel.

*Proof.* Because  $\kappa(x_{t1}, x_{t2})$  is symmetrical, it is obvious that  $v(t_1, t_2)$  is also symmetrical. But  $\kappa(x_{t1}, x_{t2})$  is also positive definite, for any set of  $N$  samples indexed by  $t$   $\{x_{t1}, \dots, x_{tN}\}$  and any set of real numbers  $\{\alpha_1, \dots, \alpha_N\}$ ,

$$\sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j \kappa(x_{t_i}, x_{t_j}) > 0. \quad (11.9)$$

It is also true that for any strictly positive function  $g(\cdot, \cdot)$  of two random variables  $x$  and  $y$ ,  $E[g(x, y)] > 0$ . Then

$$\begin{aligned} E \left[ \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j \kappa(x_{t_i}, x_{t_j}) \right] &> 0 \Rightarrow \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j E[\kappa(x_{t_i}, x_{t_j})] \\ &= \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j v(t_1, t_2) > 0. \end{aligned}$$

Thus,  $v(t_1, t_2)$  is both symmetric and positive definite. Now, the Moore-Aronszajn theorem [8] guarantees that for every real symmetric positive definite function of two real variables, there exists a unique RKHS, denoted as  $H_v$ , with the positive function as its reproducing kernel.

The RKHS  $H_v$  defined in Property 10.10 and in Property 11.6, although similar in the mapping, are defined in two different domains: the former on a pair of random variables and the latter on an index set of time instants of the random process, hence the different notations used in the cross-correntropy of Chapter 10 and the autocorrentropy function. However, to simplify the notation, both RKHS are denoted by  $H_v$  because one may consider the former a special case of the latter.

The autocorrentropy function can be interpreted in two vastly different feature spaces. One is  $H_\kappa$  induced by the Gaussian kernel on pairs of observations  $\kappa(x_{t_1}, x_{t_2})$ , which is widely used in kernel learning. The elements of this RKHS are infinite-dimensional vectors expressed by the eigenfunctions of the Gaussian kernel [288], and they lie on the positive hyperoctant of a sphere because  $\|\Phi(x)\|^2 = G_\sigma(0) = 1/\sqrt{2\pi}\sigma$ . Correntropy performs statistical averages on the functionals in this sphere. A full treatment of this view requires a differential geometry approach to take advantage of the manifold properties [211].

The second feature space is the RKHS induced by the correntropy kernel itself  $v(t_1, t_2)$  and denoted as  $H_v$ , which is defined on the index set of random variables. The inner product is defined by the correlation of the kernel at two different lags and the mapping produces a single deterministic scalar for each element on the index set, that is, the practical dimension of  $H_v$  is the size of the index set.  $H_v$  has very nice properties for statistical signal processing:

- $H_v$  provides a straightforward way to apply optimal projection algorithms based on second-order statistics that are expressed by inner products.
- The effective dimension of  $H_v$  is under the control of the designer by selecting the number of lags (just like the RKHS defined by the autocorrelation function).
- Elements of  $H_v$  can be readily manipulated algebraically for statistical inference (i.e. without taking averages over realizations).
- $H_v$  is nonlinearly related to the input space, unlike the RKHS defined by the autocorrelation of the random process (presented in Chapter 1). Therefore it is in principle very appealing for nonlinear statistical signal processing.

The problem is that we have yet to identify the Hilbert space defined from  $\{X_t, t \in T\}$  that is congruent to  $H_v$ . Obviously, the Hilbert space spanned by the set of all random variables in the linear manifold  $L(X_t, t \in T)$  (together with their limit points) is insufficient for our goal, inasmuch as the nonlinear kernel in Eq. (11.1) yields a functional that may be outside the set of all possible linear functionals on the random function  $\{X_t, t \in T\}$ . This brings difficulties in the computation with transformed functionals, and so far we have to rely on approximations using the kernel evaluated at the samples, which is only valid in the expected value sense, as we show next.

**Property 11.7.** Let  $\{x(n)\}_{n=1}^N$  be a strictly stationary random process. The autocorrentropy kernel induces a scalar nonlinear mapping  $\mu_x$  that maps the random process as  $\{\mu_x(n)\}_{n=1}^N$  while preserving the similarity measure in the sense

$$\mathbb{E}[\mu_x(n).\mu_x(n-i)] = v_\sigma(n, n-i) = \mathbb{E}[G_\sigma(x(n) - x(n-i))], \quad 0 \leq i \leq N-1. \quad (11.10)$$

Moreover, the square of the mean of the transformed data is an asymptotic estimate (as  $N \rightarrow \infty$ ) of the information potential (IP) of the original data.

*Proof.* The existence of the nonlinear mapping  $\mu_x$  for any positive definite kernel is proved in [238], and justifies the statement that positive definite kernels are covariance operators. Here we prove the second part of the property. Denote  $m_\mu$  as the mean of the transformed data,  $m_\mu = 1/N \sum_{n=1}^N \mu_x(n)$ , therefore  $m_\mu^2 = 1/N^2 \sum_{n=1}^N \sum_{i=1}^N \mu_x(n) \mu_x(i)$ .

Rewriting Eq. (11.10) in the sample estimate form, asymptotically we have

$$\sum_{n=1}^N \mu_x(n) \mu_x(n-i) = \sum_{n=1}^N G_\sigma(x(n) - x(n-i)) \quad (11.11)$$

with fixed  $i$  and as  $N \rightarrow \infty$ . We arrange the double summation of  $m_\mu^2$  as an array and sum along the diagonal direction which yield exactly the autocorrelation function of the transformed data at different lags, thus the autocorrentropy function of the input data at different lags to yield

$$\begin{aligned} \frac{1}{N^2} \sum_{n=1}^N \sum_{j=1}^N \mu_x(n) \mu_x(j) &= \frac{1}{N^2} \left( \sum_{n=i+1}^{N-1} \sum_{i=1}^N \mu_x(n) \mu_x(n-i) \right. \\ &\quad \left. + \sum_{n=1}^{N-i} \sum_{i=0}^{N-1} \mu_x(n) \mu_x(n+i) \right) \\ &\approx \frac{1}{N^2} \left( \sum_{n=i+1}^{N-1} \sum_{i=1}^N G_\sigma(x(n) - x(n-i)) \right. \\ &\quad \left. + \sum_{n=1}^{N-i} \sum_{i=0}^{N-1} G_\sigma(x(n) - x(n+i)) \right), \end{aligned} \quad (11.12)$$

which is the definition of the IP. As observed in Eq. (11.12), when the summation indices are far from the main diagonal, smaller and smaller data sizes are involved which leads to poorer approximations. Notice that this is exactly the same problem when the autocorrelation function is estimated from windowed data. As  $N$  approaches infinity, the estimation error goes to zero asymptotically. In other words, the mean of the transformed data induced by the correntropy kernel asymptotically estimates the square root of the information potential and thus Renyi's quadratic entropy of the original data. This property corroborates further the name of correntropy given to this function.

**Definition.** The generalized covariance function, called the centered autocorrentropy function is defined as

$$u(t_1, t_2) = E_{x_{t_1} x_{t_2}}[G_\sigma(x_{t_1} - x_{t_2})] - E_{x_{t_1}} E_{x_{t_2}}[G_\sigma(x_{t_1} - x_{t_2})] \quad (11.13)$$

for each  $t_1$  and  $t_2$  in  $T$ , where  $E$  denotes the statistical expectation operator and  $G_\sigma(\cdot, \cdot)$  is the Gaussian kernel. Notice that the centered autocorrentropy function is the difference between the joint expectation and the product of marginal expectations of  $G_\sigma(x_{t_1} - x_{t_2})$ .

For strictly stationary random processes the centered autocorrentropy is only a function of  $\tau = t_1 - t_2$ , and in discrete time it reads

$$u(m) = \mathbb{E}_{x_0 x_m} [G_\sigma(x(0) - x(m))] - \mathbb{E}_{x_0} \mathbb{E}_{x_m} [G_\sigma(x(0) - x(m))].$$

**Definition.** The Fourier transform of the centered autocorrentropy function of a strictly stationary process  $x(t)$ , if it exists, is called the correntropy spectral density (CSD) and is defined by

$$P_\sigma(\omega) = \int_{-\infty}^{\infty} u(\tau) e^{-j\omega\tau} d\tau. \quad (11.14)$$

$\omega$  in this expression is the frequency in radians, and a similar expression can be written for the frequency  $f$  in Hz ( $\omega = 2\pi f$ ).

So there is a new Fourier transform pair  $P(\omega) = \Im\{u(\tau)\}$ , and therefore  $u(\tau) = \Im^{-1}\{P(\omega)\}$ . This definition follows by analogy to the well-known Wiener-Kinchin theorem that establishes a relationship between the power spectrum and the Fourier transform of the autocorrelation function. This was a major result in spectral estimation, because it bypassed the fact that theoretically the Fourier transform of a random process cannot be computed. In our case, the centered autocorrentropy function is also square integrable (just as is the autocorrelation function) so its Fourier transform exists (the autocorrentropy is not square integrable because of the bias). However, the CSD is not a measure of power anymore, and it is a function of the kernel size utilized in the estimation. In particular, when the time lag  $\tau = 0$ , the correntropy spectral density function becomes

$$\begin{aligned} u(t, t) &= \mathbb{E}_{x_t x_t} [G_\sigma(x_t - x_t)] - \mathbb{E}_{x_t} \mathbb{E}_{x_t} [G_\sigma(x_t - x_t)] \\ &= \mathbb{E} [| | \Phi(x_t) | |^2] - | | \mathbb{E} [\Phi(x_t)] | |^2 = u(0) \\ &= \frac{1}{2\pi} \int_{-\infty}^{\infty} P(\omega) d\omega = \int_{-\infty}^{\infty} P(f) df \end{aligned}$$

This equation shows that the difference between the expectation of the norm square and the norm square of the expectation of the nonlinearly transformed random process is the total area under the correntropy spectral density function  $P(f)$  of the process. In traditional random process analysis, the power in the random process is the total area under the power spectral density function. By analogy, we define  $u(0)$  as the *projected power* (data is projected in the RKHS), and from a geometrical perspective,  $u(0)$  can also be considered as “generalized” variance of the nonlinearly transformed random process in the feature space. Due to the symmetry of  $u(\tau)$ , it is easy to show that the CSD is a real function and also even, just as the PSD. It is also possible to show that the CSD is nonnegative [343].

The kernel size can be interpreted as a continuous parameter that allows us to select between the conventional PSD and a spectral representation that contains different weightings of second and higher-order moments of the random

process. In fact, if we observe the Taylor series expansion of autocorrentropy (Eq. (11.3)), we see that for large kernel sizes, CSD is proportional to the PSD (Eq. (11.4)), but for values in the range given by the Silverman's rule different information is contained in CSD, because it is in fact using a different similarity metric. This may have applications in spectral estimation and detection theories, however, we and others [22] have just started to explore this relation.

### 11.3 Cross-Correntropy Function: Definition and Properties

**Definition.** Let  $\{X_t, t \in T\}$  and  $\{Y_t, t \in T\}$  be two stochastic processes with  $T$  being an index set and  $x_t, y_t \in R^d$ . The cross-correntropy function  $v_{x,y}(t_1, t_2)$  is defined as a function from  $T \times T \rightarrow R^+$  given by

$$v_{x,y;\sigma}(t_1, t_2) = E[G_\sigma(x_{t_1} - y_{t_2})]. \quad (11.15)$$

In spite of the name,  $v_{x,y}(t_1, t_2)$  is more similar to the autocorrentropy function  $v_{x,x}(t_1, t_2)$  than to the cross-correntropy  $v_\sigma(X, Y)$  defined in Chapter 10. It is also defined on an index set but now it involves two different random processes. Although similar to the autocorrentropy function, note that the cross-correntropy function is not necessarily symmetric with respect to the lag  $\tau = t_1 - t_2$ , hence it is not guaranteed to be a positive definite function. There is also a problem of the potential difference in the dynamic range of the random variables that affects the definition of the kernel. Therefore, either prescaling of the variables to the same range or ellipsoidal kernels are required.

For discrete random processes Eq. (11.15) becomes  $v_{x,y}(n, l) = E[G_\sigma(x(n) - y(n-l))]$ , which reduces to  $v_{x,y}(l)$  for strictly stationary random processes.

**Property 11.8.** The cross-correntropy function of strictly stationary random processes at lag zero (i.e.,  $v_{x,y}(0)$ ) equals the cross-correntropy for the random variables  $X$  and  $Y$  (i.e.,  $v_\sigma(X, Y)$ ).

*Proof.* In fact for a strictly stationary random process, statistical properties do not depend on the time index  $n$ , so the correntropy function can be written  $v_{x,y}(l) = E[G_\sigma(x(0) - y(-l))]$ . If we make  $l = 0$ , than  $v_{x,y}(0) = E[G_\sigma(x(0) - y(0))]$  which means that we are simply working with two random variables  $X$  and  $Y$ , and it defaults to the cross-correntropy definition of Eq. (10.2). This result helps us understand better the relation between the concepts in Chapter 10 and in this chapter.

**Property 11.9.** A necessary condition for two strictly stationary stochastic processes to be independent is:

$$v_{xy}(l) = v_{xy}(0), \quad \text{for all } l; \quad (11.16)$$

that is, the cross-correntropy function becomes a quantity that is invariant with respect to the lag  $l$ .

*Proof.* Using the RKHS notation and assuming independence and strict stationarity of  $x(n)$  and  $y(n)$ , we have the cross-correntropy function at lag  $l$  as

$$\begin{aligned} v_{x,y}(l) &= \mathbb{E}[\langle \Phi(x(n), \Phi(y(n-l)) \rangle_{H_\kappa}] = \langle \mathbb{E}[\Phi(x(n))], \mathbb{E}[\Phi(y(n-l))] \rangle_{H_\kappa} \\ &= \langle \mathbb{E}[\Phi(x(n))], \mathbb{E}[\Phi(y(n))] \rangle_{H_\kappa} = \mathbb{E}[G_\sigma(x(n) - y(n))] = v(0) \end{aligned} \quad (11.17)$$

This property is a generalization of Property 11.5 that was proved for the autocorrentropy of Gaussian processes, and it can be used as a contrast function for blind source separation of sources with or without a time structure as we show below.

The properties involving the autocorrentropy and cross-correntropy functions are far from complete, and similar results as the ones obtained in Chapter 10 for dependency may be achievable, which will extend the toolkit even further for time series analysis.

## 11.4 Optimal Linear Filters in $H_v$

In this section we illustrate how one can design optimal filters in the RKHS created by the autocorrentropy defined on the index set  $T \times T$  (denoted as  $H_v$ ). Let us design an optimal nonlinear filter trained with a set of images taken from a given object across orientations, with the intent of recognizing if a test image belongs to the object. This also illustrates the fact that random processes exist in domains other than time; in this special case the index set consists of image pixel shifts. The idea is to take the minimum average correlation energy filter (MACE) which is an optimal linear filter constructed in the image domain and map it to  $H_v$ , yielding a nonlinear filter we call CMACE [166]. We illustrate how the equations are mapped from the image domain to  $H_v$  and how they can be computed directly from data, using the approximation outlined in property 11.8. Because  $H_v$  is different from  $H_\kappa$  we believe that this exercise is useful.

Let us consider a two-dimensional image as a  $d \times 1$  column vector by lexicographically reordering the image, where  $d$  is the number of pixels, and denote the  $i$ th image vector by  $\mathbf{x}_i$  after reordering. The conventional MACE filter is better formulated in the frequency domain. The discrete Fourier transform (DFT) of the column vector  $\mathbf{x}_i$  is denoted by  $\mathbf{X}_i$  and we define the training image data matrix  $\mathbf{X}$  as  $\mathbf{X} = [\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N]$ , where the size of  $\mathbf{X}$  is  $d \times N$  and  $N$  is the number of training images. Let the vector  $\mathbf{h}$  be the  $[d \times 1]$  filter in the space domain and represent by  $\mathbf{H}$  its Fourier transform vector. We are interested in the correlation between the input image and the filter. The correlation of the  $i$ th image sequence  $x_i(n)$  with the filter impulse

response  $h(n)$  can be written as  $g_i(n) = x_i(n) \otimes h(n)$ . By Parseval's theorem, the correlation energy of the  $i$ th image can be written as a quadratic form

$$E_i = \mathbf{H}^H \mathbf{D}_i \mathbf{H}, \quad (11.18)$$

where  $E_i$  in this case represents correlation energy of the  $i$ th image,  $\mathbf{D}_i$  is a diagonal matrix of size  $d \times d$  whose diagonal elements are the magnitude squared of the associated element of  $\mathbf{X}_i$ , that is, the power spectrum of  $x_i(n)$ , and the superscript  $H$  denotes the Hermitian transpose. The objective of the MACE filter is to minimize the average output correlation energy over the image class while simultaneously satisfying an intensity constraint at the origin of the output space for each image. The value of the correlation at the origin of the output space can be written as  $g_i(0) = \mathbf{X}_i^H \mathbf{H} = c_i$ , for  $i = 1, \dots, N$  training images, where  $c_i$  is a user-specified output correlation value at the origin for the  $i$ th image. Our goal is to determine  $\mathbf{H}$ . Then the average energy over all training images is expressed as  $E_{avg} = \mathbf{H}^H \mathbf{D} \mathbf{H}$ , where  $\mathbf{D} = \frac{1}{N} \sum_{i=1}^N \mathbf{D}_i$ . The MACE design problem is to minimize  $E_{avg}$  while satisfying the constraint,  $\mathbf{X}^H \mathbf{H} = \mathbf{c}$ , where  $\mathbf{c} = [c_1, c_2, \dots, c_N]$  is an  $N$ -dimensional vector. This optimization problem can be solved using Lagrange multipliers, and the close form solution is [209]

$$\mathbf{H} = \mathbf{D}^{-1} \mathbf{X} (\mathbf{X}^H \mathbf{D}^{-1} \mathbf{X})^{-1} \mathbf{c}. \quad (11.19)$$

It is clear that the spatial filter  $\mathbf{h}$  can be obtained from  $\mathbf{H}$  by an inverse DFT. Once  $\mathbf{h}$  is determined, we apply an appropriate threshold to the output correlation plane and decide whether the test image belongs to the template's class.

## A RKHS Perspective of the MACE filter

The MACE filter can also be understood by the theory of Hilbert space representations of random functions proposed by Parzen [238]. Here we need to use a complex-valued kernel. Let  $H$  be a Hilbert space of functions on some set  $I$ , define an inner product  $\langle \cdot, \cdot \rangle_H$  in  $H$  and a complex-valued bivariate function  $\kappa(x_s, x_t)$  on  $I \times I$ . It is well known that the autocorrelation function  $R$  is nonnegative definite, therefore it determines a unique RKHS  $H_R$ , according to the Moore–Aronszajn theorem. By the Mercer's theorem [217],

$$R(i, j) = \sum_{k=0}^{\infty} \lambda_k \varphi_k(i) \varphi_k(j),$$

where  $\{\lambda_k, k = 1, 2, \dots\}$  and  $\{\varphi_k(i), k = 1, 2, \dots\}$  are a sequence of non-negative eigenvalues and corresponding normalized eigenfunctions of  $R(i, j)$ , respectively. One may define a congruence  $C$  from  $H_R$  onto the linear space  $L_2(x_i, i \in I)$  such that

$$C(R(i, \cdot)) = x_i. \quad (11.20)$$

The congruence  $C$  can be explicitly represented as

$$C(f) = \sum_{k=0}^{\infty} a_k \xi_k, \quad (11.21)$$

where  $\xi_k$  is a set of orthogonal random variables belonging to  $L_2(\varphi(i), i \in I)$  and  $f$  is any element in  $H_R$  in the form of  $f(i) = \sum_{k=0}^{\infty} \lambda_k a_k \varphi_k(i)$  and every  $i$  in  $I$ .

Let us consider the case of one training image and construct the following matrix

$$\mathbf{U} = \begin{pmatrix} x(d) & 0 & \cdots & 0 & 0 \\ x(d-1) & x(d) & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x(1) & x(2) & \cdots & \cdots & x(d) \\ 0 & x(1) & x(2) & \cdots & x(d-1) \\ 0 & 0 & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & \cdots & x(1) \end{pmatrix}, \quad (11.22)$$

where the dimension of matrix  $\mathbf{U}$  is  $(2d - 1) \times d$ . Here we denote the  $i$ th column of the matrix  $\mathbf{U}$  as  $U_i$ . Then the column space of  $\mathbf{U}$  is

$$L_2(\mathbf{U}) = \left\{ \sum_{i=1}^d \alpha_i U_i \mid \alpha_i \in R, i = 1, \dots, d \right\}, \quad (11.23)$$

which is congruent to the RKHS induced by the autocorrelation kernel

$$R(i, j) = \langle U_i, U_j \rangle = U_i^T U_j, \quad i, j = 1, \dots, d, \quad (11.24)$$

where  $\langle \cdot, \cdot \rangle$  represents the inner product operation. As long as the columns in  $\mathbf{U}$  are linearly independent, all the vectors in this space can be expressed as a linear combination of the column vectors. The optimization problem of the MACE is to find a vector  $\mathbf{g}_o = \sum_{i=1}^d h_i U_i$  in  $L_2(\mathbf{U})$  space with coordinates  $\mathbf{h} = [h_1 h_2 \cdots h_d]^T$  such that  $\mathbf{g}_o^T \mathbf{g}_o$  is minimized subject to the constraint that the  $d$ th component of  $\mathbf{g}_o$  (which is the correlation at zero lag) is some constant. Formulating the MACE filter from this RKHS viewpoint will not bring additional performance advantage over Eq. (11.19), however, it will help us derive a nonlinear extension to the MACE with correntropy.

## 11.5 Correntropy MACE (CMACE) Filter in $H_v$

According to the RKHS perspective of the MACE filter, we can extend it immediately to  $H_v$ . Applying the autocorrentropy concept to the MACE formulation of Section 11.3, the definition of the autocorrelation in Eq. (11.24) can be substituted by the autocorrentropy function

$$v(i, j) = \frac{1}{2d-1} \sum_{n=1}^{2d-1} G_\sigma(\mathbf{U}_{in} - \mathbf{U}_{jn}) \quad i, j = 1, \dots, d \quad (11.25)$$

where  $\mathbf{U}_{in}$  is the  $(i, n)$  th element in Eq. (11.22). This function is positive definite and thus induces the  $H_v$ . According to Mercer's theorem, there is a basis  $\{\mu_i, i = 1, \dots, d\}$  in this  $H_v$  such that

$$\langle \mu_i, \mu_j \rangle = v(i, j) \quad i, j = 1, \dots, d \quad (11.26)$$

Because it is a  $d$  dimensional Hilbert space, it is isomorphic to any  $d$ -dimensional real vector space equipped with the standard inner product structure. After an appropriate choice of this isomorphism  $\{\mu_i, i = 1, \dots, d\}$ , which is nonlinearly related to the input space, a nonlinear extension of the MACE filter can be readily constructed, namely, finding a vector  $\mathbf{v}_0 = \sum_{i=1}^d f_{h,i} \mu_i$  with  $\mathbf{f}_h = [f_{h1} \cdots f_{hd}]^T$  as coordinates such that  $\mathbf{v}_0^T \mathbf{v}_0$  is minimized subject to the constraint that the  $d$ th component of  $\mathbf{v}_0$  is some prespecified constant.

Let the  $i$ th image vector be  $\mathbf{x}_i = [x_i(1) x_i(2) \cdots x_i(d)]^T$  and the filter be  $\mathbf{h} = [h(1) h(2) \cdots h(d)]^T$ , where T denotes transpose. From Property 11.7 with the time index substituted by the row index, the CMACE filter can be formulated in feature space by applying a nonlinear mapping function  $f$  onto the data as well as the filter. We denote the transformed training image matrix of size  $d \times N$  by

$$\mathbf{F}_X = [\mathbf{f}_{x_1}, \mathbf{f}_{x_2}, \dots, \mathbf{f}_{x_N}], \quad \mathbf{f}_{x_i} = [f(x_i(1)) f(x_i(2)) \cdots f(x_i(d))]^T, \quad i = 1, \dots, N$$

and the transformed filter vector  $\mathbf{f}_h = [f(h(1)) f(h(2)) \cdots f(h(d))]^T$ , of size  $d \times 1$ . Given data samples, the cross-correntropy between the  $i$ th training image vector and the filter can be estimated as

$$v_{oi}[m] = \frac{1}{d} \sum_{n=1}^d f(h(n)) f(x_i(n-m)), \quad (11.27)$$

for all the lags  $m = -d+1, \dots, d-1$ . Then the cross-correntropy vector  $\mathbf{v}_{oi}$  is formed with all the lags of  $v_{oi}(m)$  denoted by  $\mathbf{v}_{oi} = \mathbf{S}_i \mathbf{f}_h$ , where  $\mathbf{S}_i$  is the matrix of size  $(2d-1) \times d$  as

$$\mathbf{S}_i = \begin{pmatrix} f(x_i(d)) & 0 & \cdots & 0 & 0 \\ f(x_i(d-1)) & f(x_i(d)) & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f(x_i(1)) & f(x_i(2)) & \cdots & \cdots & f(x_i(d)) \\ 0 & f(x_i(1)) & f(x_i(2)) & \cdots & f(x_i(d-1)) \\ 0 & 0 & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & f(x_i(1)) & f(x_i(2)) \\ 0 & 0 & \cdots & 0 & f(x_i(1)) \end{pmatrix}. \quad (11.28)$$

The scale factor  $1/d$  has no influence on the solution, therefore it is ignored. The correntropy energy of the  $i^{\text{th}}$  image is given by

$$E_i = \mathbf{v}_{oi}^T \mathbf{v}_{oi} = \mathbf{f}_h^T \mathbf{S}_i^T \mathbf{S}_i \mathbf{f}_h, \quad (11.29)$$

Denoting  $\mathbf{V}_i = \mathbf{S}_i^T \mathbf{S}_i$  and using the definition of autocorrentropy in Eq. (11.2), the  $d \times d$  correntropy matrix  $\mathbf{V}_i$  is

$$\mathbf{V}_i = \begin{pmatrix} v_i(0) & v_i(1) & \cdots & v_i(d-1) \\ v_i(1) & v_i(0) & \cdots & v_i(d-2) \\ \vdots & \vdots & \ddots & \vdots \\ v_i(d-1) & \cdots & v_i(1) & v_i(0) \end{pmatrix},$$

where each element of the matrix is computed without explicit knowledge of the mapping function  $f$  by

$$v_i(l) = \sum_{n=1}^d G_\sigma(x_i(n) - x_i(n+l)) \quad (11.30)$$

for  $l = 0, \dots, d-1$ . The average correntropy energy over all the training data is

$$E_{av} = \frac{1}{N} \sum_{i=1}^N E_i = \mathbf{f}_h^T \mathbf{V}_X \mathbf{f}_h, \quad (11.31)$$

where,  $\mathbf{V}_X = \frac{1}{N} \sum_{i=1}^N \mathbf{V}_i$ . Because our objective is to minimize the average correntropy energy in feature space, the optimization problem is reformulated as

$$\min \mathbf{f}_h^T \mathbf{V}_X \mathbf{f}_h \text{ subject to } \mathbf{F}_X^T \mathbf{f}_h = \mathbf{c}, \quad (11.32)$$

where,  $\mathbf{c}$  is the desired vector for all the training images. The constraint in Eq. (11.32) means that we specify the correntropy values between the training input and the filter as the desired constant. Because the correntropy matrix  $\mathbf{V}_X$  is positive definite, there exists an analytic solution to the optimization problem using the method of Lagrange multipliers in the new finite-dimensional  $H_v$ . Then the CMACE filter in feature space becomes

$$\mathbf{f}_h = \mathbf{V}_X^{-1} \mathbf{F}_X \left( \mathbf{F}_X^T \mathbf{V}_X^{-1} \mathbf{F}_X \right)^{-1} \mathbf{c}. \quad (11.33)$$

The transformation from the input space to  $H_v$  is nonlinear and the inner product structure of  $H_v$  provides the possibility of obtaining closed-form optimal nonlinear filter solutions utilizing second- and high-order statistics.

Another important difference compared with existing machine learning methods based on the conventional kernel method, which normally yields an infinite-dimensional feature space, is that  $H_v$  has the same effective dimension as the input space. The goal of this new algorithm is to find a template  $\mathbf{f}_h$  in this  $H_v$  such that the cost function is minimized subject to the constraint. Therefore, the number of degrees of freedom of this optimization problem is

still  $d$ , so regularization, which will be needed in traditional kernel methods, is not necessary here. The dimensionality fixed by the image domain also carries disadvantages because the user has no control of  $H_v$ 's effective dimensionality. Therefore, the effectiveness of the nonlinear solution depends solely on the nonlinear transformation between the input space and  $H_v$ . The theoretical advantage of using this feature space is justified by the CIM metric, which is very suitable to quantify similarity in feature spaces and should improve the robustness to outliers of the conventional MACE.

### CMACE Filter Implementation

Because the nonlinear mapping function  $f$  is not explicitly known, it is impossible to directly compute the CMACE filter  $\mathbf{f}_h$  in the feature space. However, the correntropy output can be obtained by the inner product between the transformed input image and the CMACE filter in  $H_v$ . In order to test this filter, let  $\mathbf{Z}$  be the matrix of testing images (each a length- $L$  vector) and  $\mathbf{F}_Z$  be the transformed matrix of  $\mathbf{Z}$ ; then the  $L \times 1$  output vector is given by

$$\mathbf{y} = \mathbf{F}_Z^T \mathbf{V}_X^{-1} \mathbf{F}_X \left( \mathbf{F}_X^T \mathbf{V}_X^{-1} \mathbf{F}_X \right)^{-1} \mathbf{c}. \quad (11.34)$$

Here, we denote  $\mathbf{K}_{ZX} = \mathbf{F}_Z^T \mathbf{V}_X^{-1} \mathbf{F}_X$  and  $\mathbf{K}_{XX} = \left( \mathbf{F}_X^T \mathbf{V}_X^{-1} \mathbf{F}_X \right)^{-1}$ . Then the output becomes

$$\mathbf{y} = \mathbf{K}_{ZX} (\mathbf{K}_{XX})^{-1} \mathbf{c}, \quad (11.35)$$

where  $\mathbf{K}_{XX}$  is an  $N \times N$  symmetric matrix and  $\mathbf{K}_{ZX}$  is an  $L \times N$  matrix whose  $(i, j)$ th elements are expressed by

$$\begin{aligned} (\mathbf{K}_{XX})_{ij} &= \sum_{l=1}^d \sum_{k=1}^d w_{lk} f(x_i(k)) f(x_j(l)) \cong \sum_{l=1}^d \sum_{k=1}^d w_{lk} G_\sigma(x_i(k) - x_j(l)), \\ i, j &= 1, \dots, N \\ (\mathbf{K}_{ZX})_{ij} &= \sum_{l=1}^d \sum_{k=1}^d w_{lk} f(z_i(k)) f(x_j(l)) \cong \sum_{l=1}^d \sum_{k=1}^d w_{lk} G_\sigma(z_i(k) - x_j(l)), \\ i &= 1, \dots, L, j = 1, \dots, N \end{aligned} \quad (11.36)$$

where  $w_{lk}$  is the  $(l, k)$ th element of  $\mathbf{V}_X^{-1}$ .

The final computable output in Eq. (11.36) is obtained by approximating  $f(x_i(k))f(x_j(l))$  and  $f(z_i(k))f(x_j(l))$  by  $G_\sigma(x_i(k) - x_j(l))$  and  $G_\sigma(z_i(k) - x_j(l))$  respectively, which hold on average because of Property 11.7.

The CMACE, although a nonlinear filter in the input space, preserves the shift-invariant property of the linear MACE [209]. The output of the CMACE in Eq. (11.35) is only one value for each input image, but it is possible to construct the whole output plane by shifting the test input image and as a

result, the shift invariance property of the correlation filters can be utilized at the expense of more computation. Applying an appropriate threshold to the output of Eq. (11.35), one can detect and recognize the testing data without generating the composite filter in feature space.

### Centering the CMACE in $\mathbf{H}_v$

With the Gaussian kernel, the correntropy value is always positive, which brings the need to subtract the mean of the transformed data in feature space in order to suppress the effect of the output DC bias. Therefore, the centered correntropy matrix  $\mathbf{V}_{XC}$  can be used in the CMACE output Eq. (11.35), by subtracting the information potential from the correntropy matrix  $\mathbf{V}_X$  as

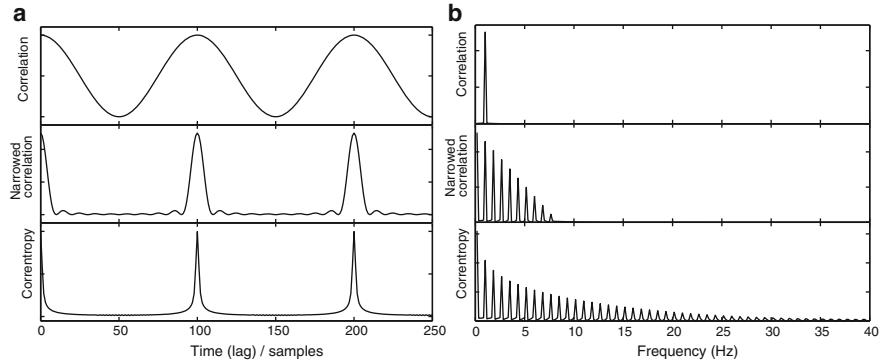
$$\mathbf{V}_{XC} = \mathbf{V}_X - m_{favg}^2 \cdot \mathbf{1}_{d \times d}, \quad (11.37)$$

where  $m_{favg}^2$  is the average estimated information potential over  $N$  training images and  $\mathbf{1}_{d \times d}$  is a  $d \times d$  matrix with all the entries equal to 1. Using the centered correntropy matrix  $\mathbf{V}_{XC}$ , a better rejection ability for out-of-class images is achieved.

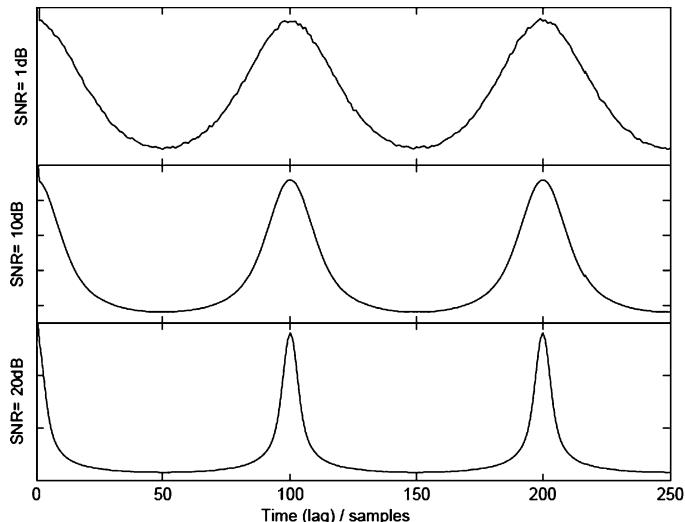
## 11.6 Application: Autocorrentropy Function as a Similarity Measure over Lags

Correlation is widely used to measure similarity of time series. Perhaps the most obvious example is to compare the similarity produced by autocorrentropy with the one produced by the autocorrelation function. As is well-known, the autocorrelation function of a sinewave is a sinewave. This is such a well known fact in signal processing that we do not even ask what the similarity should look like. If one recalls how the autocorrelation is obtained (shifting one replica of the signal over the original signal) we dare to suggest that the similarity should be a very peaky signal every time the two versions of the sinewave align perfectly. Of course this is not what is observed with autocorrelation (Figure 11.1 top row).

However, when the autocorrentropy function is computed for the sinewave with a Gaussian kernel with size selected according to Silverman's rule, we obtain the plot of Figure 11.1, bottom row. This seems to be a more "intuitive" measure of similarity because it emphasizes clearly the periodicity! It is easy to understand why this is if we look at Eq. (11.2). Notice that irrespective of the amplitude of the samples, when they are similar, the autocorrentropy assigns a value close to 1, while the autocorrelation multiplies the amplitudes, de-emphasizing the contributions of the low-amplitude samples. Obviously, we are using two different similarity metrics when computing the two measures (see CIM in Chapter 3). Because of the narrowness of correntropy functions in the time domain, we anticipate the rich harmonics present in the frequency



**Fig. 11.1.** (a) Autocorrelation function of a sinusoid (top figure), narrowband autocorrelation (with product of ten lags) in the middle and autocorrentropy of a sinusoidal signal of period 1s with 100 Hz sampling frequency (bottom figure). Kernel size is 0.04 (Silverman's rule). (b) Corresponding Fourier transforms (from [342]).



**Fig. 11.2.** Autocorrentropy for the sinewave with decreasing levels of additive white noise (SNR = 1 dB, 10 dB, and 20 dB respectively) (from [342]).

domain, as we see in Figure 11.1b. For this reason, the lag domain is the most obvious to apply correntropy, but the richness of the harmonics in the CSD may lead to many unforeseen frequency domain applications. Of course, noise in the signal will affect the peakiness of the autocorrentropy because even when the two signals are aligned the result may be away from zero. Hence, as the noise increases the peak value tends to decrease, and broaden the peak as shown in Figure 11.2. Note that in the high noise case (1 dB), the solution for the autocorrentropy function approaches the correlation function.

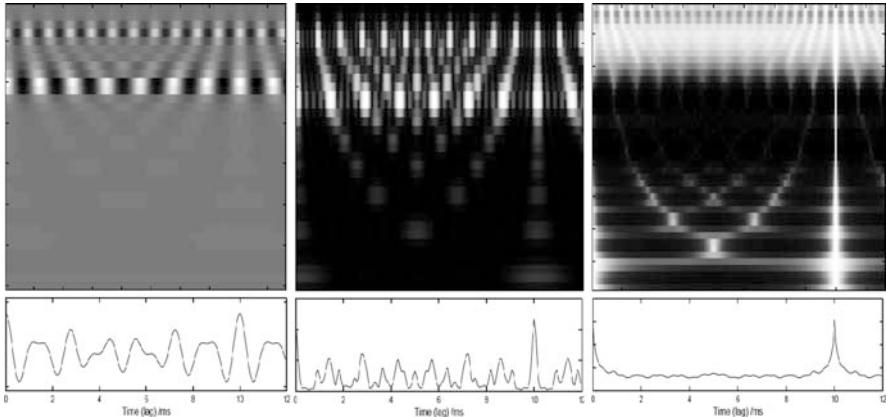
## Pitch Determination Algorithms for Speech

Pitch in speech processing is a valuable descriptor for speaker identification because it is related to the vibration of the vocal cords, which is an individual attribute. The pitch determination algorithm (PDA) described here uses cochlear filtering to pre-process the speech signal. This is achieved by a bank of 64 gammatone filters distributed in frequency according to their bandwidths [242]. The impulse response of a gammatone filter is defined as  $q(t) = t^{n-1} e^{-2\pi at \cos(2\pi f_0 t + \psi)}$ , where  $n$  is the filter order with center frequency at  $f_0$  Hz,  $\Psi$  is the phase, and  $a$  is the bandwidth parameter. The bandwidth increases quasi-logarithmically with respect to the center frequency. The center frequencies of each filter are equally spaced on the equivalent rectangular bandwidth scale between 80 and 4000 Hz. This creates a cochleogram, which has served as a biologically preprocessing in computational auditory scene analysis (CASA) [329], and is used extensively in pitch determination [147].

The analysis is done by computing the autocorrentropy function at the output of each cochlear frequency channel,  $v_i(\tau) = 1/N \sum_{n=0}^{N-1} G_\sigma(x_n^i, x_{n-\tau}^i)$  where the subscript  $i$  stands for channel number of the cochlear output. The kernel bandwidth is determined using Silverman's rule [300]. The time lag  $\tau$  is chosen long enough to include the lowest expected pitch (10 ms). Gray coding the amplitudes of the autocorrentropy for each cochlear channel in the y-axis as a function of the correntropy lags in the horizontal axis forms a picture (white high, black low). We name it correntropy-gram, which literally means “picture of correntropy”. If a signal is periodic, strong vertical lines at certain correntropy lags appear in the correntropy-gram indicating times when a large number of cochlear channels display synchronous oscillations in correntropy. The horizontal banding signifies different amounts of energy across frequency regions. The correntropy-gram is similar to the correlogram in structure but different in content because it does not display power, but “instantaneous” information potential. In order to reduce the dynamic range for display in the correntropy-gram, the correntropy function should be normalized such that the zero lag value is one as given by Eq. (11.13).

In order to emphasize pitch-related structure in the correntropy-gram, the correntropy functions are summed up across all the channels to form a “pooled” or “summary” correntropy-gram,  $S(\tau) = \sum_i u_i(\tau)$ . The summary correntropy-gram measures how likely the pitch would be perceived at a certain time lag, and the pitch frequency can be obtained by inverting the time lag. In Figure 11.3,  $S(\tau)$  is first normalized by subtracting the mean and dividing by the maximum absolute value. The position of pitch can be found by various peak-picking algorithms to identify local maxima above the predefined threshold.

We present a simple experiment to validate our method, by comparing the correntropy-gram with the conventional autocorrelation function and the narrowed autocorrelation function [42] in determining pitches for single speakers.



**Fig. 11.3.** Comparison of the correlogram, narrow autocorrelogram ( $L = 15$ ) and the correntropy-gram (Gaussian kernel and  $s = 0.01$ ) for the synthetic vowel /a/ (from [342]).

The synthetic vowels are produced by Slaney’s Auditory Toolbox [34]. For a fair comparison, we did not apply any postprocessing to the correlogram.

Figure 11.3 presents the pitch determination results for a single synthetic vowel /a/ with 100 Hz fundamental frequency. The upper plots are the images of correlation functions, narrowed autocorrelations, and correntropy functions using the same cochlear model, respectively. The bottom figures are the summaries of those three images. The kernel size in the Gaussian kernel is 0.01 and  $L = 15$  lags is used in the narrowed autocorrelation function. The conventional autocorrelation and narrowed autocorrelation are all able to produce peaks at 10 ms corresponding to the pitch of the vowel. But they also generate other erroneous peaks that might confuse pitch determination in normal speech. On the contrary, the summary of correntropy-gram provides only one single and narrow peak at 10 ms which is the pitch period of the vowel sound. And the peak is much narrower than those obtained from other methods.

In [342] we show ROC plots to compare the detectability of these methods for one, two, and three synthetic vowels using thresholds and neural networks and show the robust performance achieved with autocorrentropy. Furthermore, implementation complexity is relatively small. However, the proper selection of the kernel size is still an open issue.

We also tested our pitch determination algorithm in Bagshaw’s database [15], which contains 7,298 male and 16,948 female speech samples. The ground truth pitch is estimated at reference points based on laryngograph data. These estimates are assumed to be equal to the perceived pitch. The signal is segmented into 38.4 ms duration centered at the reference points in order to make the comparisons between different PDAs fair. The sampling frequency is 20 kHz. The kernel size is selected according to Silverman’s rule for different segments. We calculate the normalized correntropy function to yield unit

**Table 11.1.** PDAS Gross Error (%) (From [342])

PDA	Male		Female		Weighted Mean (%)
	High (%)	Low (%)	High (%)	Low (%)	
HPS	5.34	28.20	0.46	1.61	11.54
SRPD	0.62	2.01	0.39	5.56	4.95
CPD	4.09	0.64	0.61	3.97	4.63
FBPT	1.27	0.64	0.60	3.35	3.48
IPTA	1.40	0.83	0.53	3.12	3.22
PP	0.22	1.74	0.26	3.20	3.01
SHR	1.29	0.78	0.75	1.69	2.33
SHAPE	0.95	0.48	1.14	0.47	1.55
eSRPD	0.90	0.56	0.43	0.23	0.90
Correntropy	0.71	0.42	0.35	0.18	0.71

value at zero lag. Because the pitch range for a male speaker is 50–250 Hz and 120–400 Hz for a female speaker, the PDA searches local maxima from 2.5 ms to 20 ms in the summary correntropy function. We set the threshold to be 0.3 by trial and error so that every local maximum that exceeds 0.3 will be detected as a pitch candidate.

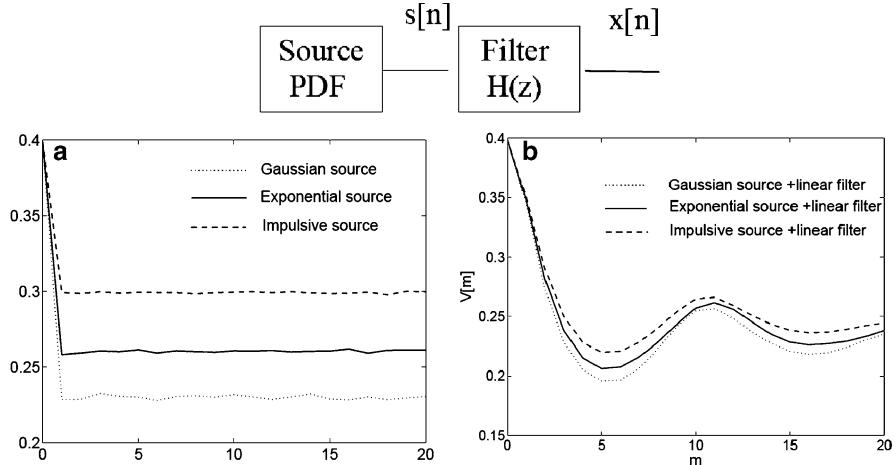
Table 11.1 summarizes the performance of various PDAs taken from [15]. The performance criterion is the relative number of gross errors. A gross error occurs when the estimated fundamental frequency is more than 20% off the true pitch value.

The percent gross errors by gender and by lower or higher pitch estimates with respect to the reference are given in Table 11.1. The weighted gross error is calculated by taking into account the number of pitch samples for each gender. It clearly shows that for this particular database correntropy-based PDA outperforms others.

### Time Series Analysis with Autocorrentropy

We consider the generative model shown in Figure 11.4, where different source distributions and different linear time-invariant filters can be used.

First, we switch off the filter to obtain  $s(n)$  and generate zero-mean unit variance white processes with different distributions: Gaussian, impulsive, and exponential. The correntropy function has been estimated from  $N = 10,000$  samples, using a Gaussian kernel with  $\sigma = 1$ . Figure 11.4a shows that the correntropy for zero lag is independent of the data and depends only on the kernel size, which is guaranteed as shown in Property 11.3. For lag  $m = 1$  the values differ for each source distribution, and these values are basically unchanged across lags as predicted by Eq. (11.5). Specifically, the value of autocorrentropy at any nonzero lag corresponds to the estimation of the information potential, which can be theoretically obtained for the Gaussian (0.23), impulsive (0.30), and exponential sources (0.26). Also recall that, for i.i.d. data, the



**Fig. 11.4.** Data generative model (top). (a) autocorrentropy for white sources  $s(n)$  and (b) linear filtered sources  $x(n)$  (from [283]).

correntropy estimate at any lag different from zero is directly related to the estimate of quadratic Renyi's entropy. The computed autocorrelation function (not shown), is basically constant as expected for this example (the variance of the sources are equal).

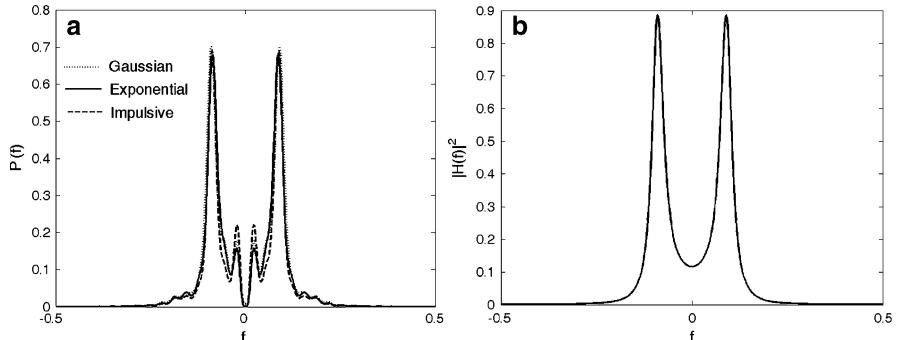
Next, the sources are filtered by the following second-order IIR filter  $H(z) = 1/(1 - 1.5z^{-1} + 0.8z^{-2})$  to obtain  $x(n)$ . Figure 11.4b shows that in this case the differences are greatly attenuated due to the Gaussianization effect of the linear filter, and the time structure created by the impulse response of the filter is readily visible in the autocorrentropy. The value of the autocorrentropy for very large lag (after the oscillations die down) is the best indicator of the differences among noise statistics.

If we compare the power spectral density of  $x(n)$  estimated from the data with the correntropy spectral density in Figure 11.5, we see that the center frequency of the filter can be inferred from both plots, but the CSD has extraneous peaks due to the peakiness of autocorrentropy. Recall that all these plots are a function of the kernel used and its parameters, so for practical use the selection of the kernel size is an important step.

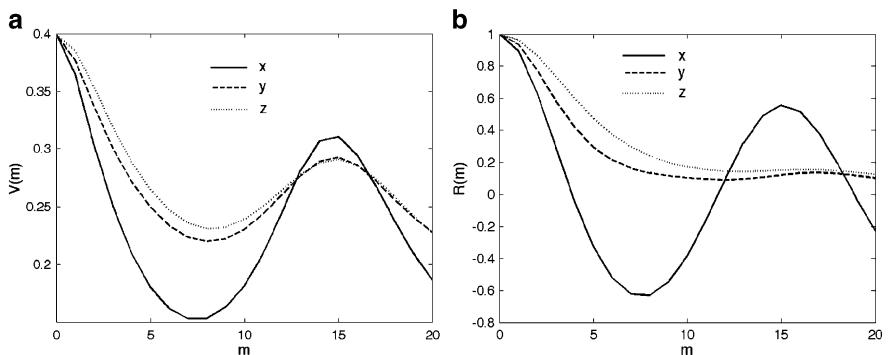
In the final example we consider the chaotic Lorenz system described as

$$\begin{aligned}\dot{x} &= a(y - x) \\ \dot{y} &= -y - xz + cx \\ \dot{z} &= xy - bz\end{aligned}$$

where  $c = 28$ ,  $a = 10$ , and  $b = 8/3$ . We generate 9000 samples of the Lorenz system by solving the equations with a fourth-order Runge-Kutta method with integral step 0.05. The goal is to compare the behavior of the autocorrentropy versus the autocorrelation in time series generated by a nonlinear



**Fig. 11.5.** Correntropy spectral density (a) and power spectral density (b) for the linear filter output (normalized digital frequency) (from [248]).



**Fig. 11.6.** Autocorrentropy (a) and autocorrelation (b) for the Lorenz chaotic system (from [248]).

model. Figure 11.6a shows that  $v(m)$  peaks at the same lag for all the state variables: because  $x$  is coupled nonlinearly to  $y$  and  $z$  the periodic similarities in one state variable affect the other states so that any state of the system can be used to gain information about the underlying nonlinear process. This does not happen with the conventional autocorrelation, which when applied to the  $y$  and  $z$  outputs shows a much decreased and hard to quantify periodicity around the fifteenth lag that is produced by the  $x$  component. This example clearly shows that autocorrentropy is able to extract nonlinear coupling information embedded in the time structure of time series, unlike the conventional autocorrelation. A simple comparison of both functions (or their spectra) may yield a statistic to distinguish nonlinear models from linear ones.

We propose  $S(\sigma) = \sum_{i=1}^N \sum_j^N (\hat{V}_\sigma(x_i) - \hat{V}_\sigma(y_i)) / (V_\sigma(0) - \text{Min } \hat{V}_\sigma(x_i))$  as a separation index to help define the proper range for the comparisons ( $\sigma = 1$  for this case).

## 11.7 Application: Karhunen-Loeve Transform in $\mathbf{H}_v$

In this case study, we present a correntropy extension to the Karhunen-Loeve transform (KLT) which is widely utilized in subspace projections [203], and is named correntropy KLT (CoKLT). Suppose the strict sense stationary discrete time signal is  $\{x(i), i = 1, 2, \dots, N + L - 1\}$  and we can map this signal as a trajectory of  $N$  points in the reconstruction space of dimension  $L$ . The data matrix over the lags is

$$\mathbf{X} = \begin{bmatrix} x(1) & x(2) & \cdots & x(N) \\ \vdots & \vdots & \ddots & \vdots \\ x(L) & x(L+1) & \cdots & x(N+L-1) \end{bmatrix}_{L \times N}. \quad (11.38)$$

The Karhunen-Loeve transform estimates the eigenfilters and principal components of the autocorrelation function or, equivalently, its principal component analysis (PCA) [253]. This sounds like PCA, however, originally the KLT was defined for continuous time signals, unlike PCA. In order to extend the KLT easily to CoKLT, let us start by writing the autocorrelation (ACM) and Gram matrices, denoted as  $\mathbf{R}$  and  $\mathbf{K}$ , respectively, as

$$\mathbf{R} = \mathbf{XX}^T \approx N \times \begin{bmatrix} r(0) & r(1) & \cdots & r(L-1) \\ r(1) & \ddots & \ddots & \vdots \\ \vdots & \ddots & r(0) & r(1) \\ r(L-1) & \cdots & r(1) & r(0) \end{bmatrix}_{L \times L} \quad (11.39)$$

$$\mathbf{K} = \mathbf{X}^T \mathbf{X} \approx L \times \begin{bmatrix} r(0) & r(1) & \cdots & r(N-1) \\ r(1) & \ddots & \ddots & \vdots \\ \vdots & \ddots & r(0) & r(1) \\ r(N-1) & \cdots & r(1) & r(0) \end{bmatrix}_{N \times N}, \quad (11.40)$$

where  $r(k) = \mathbf{E}[x(i)x(i+k)]$  is the autocorrelation function of  $x$ . When  $N$  and  $L$  are large, Eqs. (11.39) and (11.40) are both good approximations to the statistical quantities. In the following derivation, we show that  $L$  is actually not involved in the new algorithm, so we can always assume  $L$  is set appropriately to the application. Assuming  $L < N$ , by singular value decomposition (SVD) we have

$$\mathbf{X} = \mathbf{UDV}^T, \quad (11.41)$$

where  $\mathbf{U}$ ,  $\mathbf{V}$  are two orthonormal matrices and  $\mathbf{D}$  is a pseudodiagonal  $L \times N$  matrix with singular values  $\{\sqrt{\lambda_1}, \sqrt{\lambda_2}, \dots, \sqrt{\lambda_L}\}$  as its entries. Therefore,

$$\mathbf{R} = \mathbf{XX}^T = \mathbf{UDD}^T \mathbf{U}^T \quad (11.42)$$

$$\mathbf{K} = \mathbf{X}^T \mathbf{X} = \mathbf{V} \mathbf{D}^T \mathbf{D} \mathbf{V}^T. \quad (11.43)$$

From Eqs. (11.42) and (11.43), the columns of  $\mathbf{U}$  and  $\mathbf{V}$  are eigenvectors of  $\mathbf{R}$  and  $\mathbf{K}$ , respectively. We may also write Eq. (11.41) as

$$\mathbf{U}^T \mathbf{X} = \mathbf{D} \mathbf{V}^T \quad (11.44)$$

or equivalently,

$$U_i^T \mathbf{X} = \sqrt{\lambda_i} V_i^T, \quad i = 1, 2, \dots, L. \quad (11.45)$$

Here  $U_i$  and  $V_i$  are the  $i$ th columns of  $\mathbf{U}$  and  $\mathbf{V}$ , respectively. This equation simply shows that the projected data onto the  $i$ th eigenvector of  $\mathbf{R}$  is exactly the scaled  $i$ th eigenvector of  $\mathbf{K}$ . This derivation provides another viewpoint to understand why kernel PCA obtains the principal components from the Gram matrix [289]. As is well known in conventional PCA, the principal components can either be obtained by eigendecomposing the autocorrelation matrix and then projecting the data or by eigendecomposing the Gram matrix directly.

Moreover, by Property 11.7, there exists a scalar nonlinear mapping  $\mu_x(\cdot)$  (not  $\Phi$ ) that maps the input signal as  $\{\mu_x(i), i = 1, \dots, N + L - 1\}$  while preserving the similarity measure

$$\mathbb{E}[\mu_x(i) \cdot \mu_x(j)] = \mathbb{E}[G_\sigma(x(i) - x(j))]. \quad (11.46)$$

In other words, the autocorrelation function of  $\mu_x(i)$  is given by the autocorrentropy function of  $x$ . With these results, the correntropy extension to KLT is straightforward. We simply replace the autocorrelation entries with the autocorrentropy entries in Eq. (11.40) and obtain the principal components by eigendecomposition of the new Gram matrix  $\mathbf{K}$ . Therefore, CoKLT is similar to kernel PCA in the sense that it obtains the principal components by eigendecomposing the Gram matrix. However, we are talking about two entirely different RKHSs,  $H_v$  and  $H_\kappa$ , respectively.

As a practical example, we apply CoKLT to extract a single sinusoidal signal corrupted with impulsive noise  $x(i) = \sin(2\pi fi) + A \cdot z(i)$  for  $i = 1, 2, \dots, N + L - 1, \dots$ ,  $z(i)$  is a white noise process drawn from the mixture of Gaussian PDF

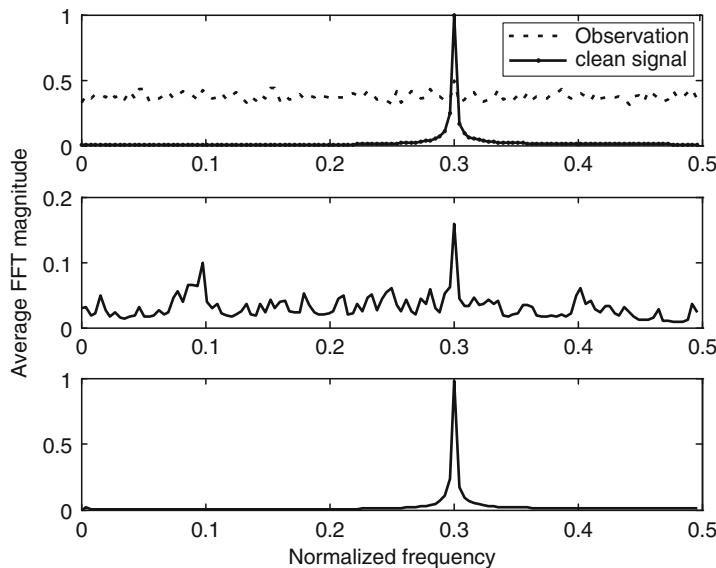
$$p_z(z) = 0.8G_{0.1}(z, 0) + 0.1G_{0.1}(z, 4) + 0.1G_{0.1}(z, -4).$$

We set  $N = 256$ ,  $f = 0.3$ , and generate  $3N$  data to estimate  $N$  point autocorrelation and correntropy functions. For a fair comparison, we choose to eigendecompose the  $N$ -by- $N$  Gram matrix for both methods. Results from eigendecomposing the  $L$ -by- $L$  autocorrelation matrix and then projecting the data are also presented for comparison. For each case in Table 11.2, 1000 Monte Carlo trials with different noise realizations are run to evaluate the performance of CoKLT and KLT. The kernel size is set to  $\sigma = 1$  in CoKLT (for finding sinusoids in noise, the kernel size can be scanned until the best line spectrum is obtained; see Figure 11.8).

For  $A = 5$ , the probability of detecting the sinusoidal signal successfully (based on using the largest peak in the spectrum) is 100% for CoKLT, compared with 15% for  $N$ -by- $N$  autocorrelation (Figure 11.7).

**Table 11.2.** Results of CoKLT (From [201])

A	CoKLT (2nd PC), %	Centering CoKLT (1st PC), %	PCA by $N$ -by- $N$ Gram $M$ ( $N = 256$ ), %	PCA by $L$ -by- $L$ ACM ( $L = 4$ ), %	PCA by $L$ -by- $L$ ACM ( $L = 30$ ), %	PCA by $L$ -by- $L$ ACM ( $L = 100$ ), %
5	100	100	15	3	4	8
4	100	100	27	6	9	17
2	100	100	99	47	73	90



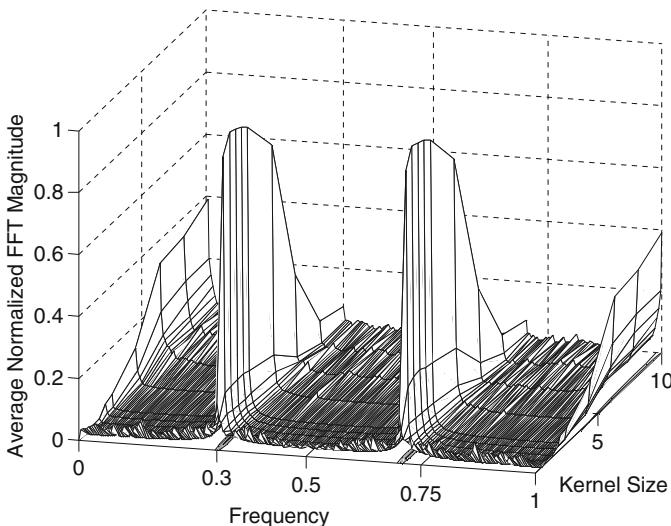
**Fig. 11.7.** CoKLT of sinusoidal signal in impulsive noise for  $A = 5$ . FFT of clean sinusoidal signal and noisy observation (top); Average of FFT of first principal component by SVD  $N$ -by- $N$  auto-correlation matrix(middle); Average of FFT of second principal component by SVD  $N$ -by- $N$  correntropy matrix (bottom) (from [201]).

The second principal component is used in CoKLT instead of the first one, because the transformed data are not centered in the feature space and the mean introduces a large DC component, which is picked as the first principal component (substituting Eq. (11.13) in Eq. (11.6) shows that the first principal component of the correntropy matrix is always a DC component [201]). Results are also shown for CoKLT when it uses a centered Gram matrix [294].

A second set of simulations is run to show how the kernel size affects the performance of CoKLT and to throw some light on how to choose it appropriately. A total of 1000 Monte Carlo simulations is run for each kernel size. The results for CoKLT (with  $A = 5$ ) are listed in Table 11.3.

**Table 11.3.** The Effects of the Kernel Size on CoKLT (From [201])

Kernel size	Centering CoKLT (%)
0.1	48
0.5	93
1.0	100
1.5	99
2.0	98
3.0	95
3.5	90
4.0	83
8.0	10

**Fig. 11.8.** Average of normalized FFT magnitude of first principal component of centering correntropy matrix over the change of kernel size (from [201]).

A graphical depiction of Table 11.3 is achieved by plotting the correntropy spectral density (i.e., the Fourier transform of the correntropy function) as a function of the kernel size (Figure 11.8).

The  $y$ -axis shows the average normalized amplitude across the 1,000 runs, which can be interpreted as the percentage of time that the particular frequency has the largest amplitude. We see that for kernel sizes between 1 and 2, the largest peak consistency occurred at the frequency corresponding to the sinusoid. The FFT magnitude for a kernel size of 10 is very similar to the power spectrum of the data. In this latter case, the highest peak in the spectrum corresponded to the sinusoid in only 10% of the runs. We tried Kernel PCA [288] on this problem and the results were disappointing. When the Gaussian kernel with large kernel size is employed, kernel PCA is almost equivalent to linear PCA but not better.

An explanation of the behavior of CoKLT is provided by Property 10.9. The sinusoidal signal in this example has a relatively small dynamic range and mainly clusters in the Euclidean zone of the CIM space (refer to Chapter 3 for a discussion of the CIM metric). The kernel size to implement this condition can always be obtained as long as we have sufficient data. The impulsive noise produces outliers in the rectification zone. CoKLT employs two steps: forward nonlinear transformation and backward linear approximation. In the forward nonlinear transformation, the method discards the large outliers moving the “bad” points across the contours in CIM space, and emphasizes the correlation of the sinewave that falls into the Euclidean zone; then by implicit backward linear approximation, the method simply treats correntropy entries as second-order statistics. This perspective justifies our correntropy extension to second-order methods and explains why this method works well in impulsive noise environments.

This insight also provides a clue on how to choose an appropriate kernel size for this application. Namely, the kernel size should be chosen such that the normal signal dynamics lie in the Euclidean zone and the impulsive outliers are kept in the rectification zone. Indeed, the sinusoidal peak-to-peak amplitude in this example is about 2 and the impulsive outliers are above 8, so an appropriate kernel size is in the range of [0.5, 3.5] according to the  $3\sigma$  condition for outlier rejection and the variance condition of Eq. (10.19).

In a sense, Figure 11.8 exemplifies a new type of frequency analysis based on correntropy, where a single parameter (the kernel size) in the correntropy function is able to scan between the conventional linear frequency analysis (large kernel sizes) and a nonlinear (higher-order moments) frequency analysis (smaller kernel sizes). However, we have to say that the realistic case of finding sinewaves in noise is much more complicated than this example because there are normally many sinewaves present of different amplitudes. Although it may still be advantageous to use correntropy, further research is necessary to analyze the structure of the signal and noise spaces.

## 11.8 Application: Blind Source Separation

Here we use the very interesting properties of the cross-correntropy that blend time and statistical information to propose two new contrast functions (criteria) for BSS: one using both the time and statistical structure of the data obtained from the cross-correntropy function,  $v_{x,y}(t_1, t_2)$ , and the other using only statistical information gathered from the cross-correntropy,  $v_\sigma(X, Y)$ .

### Contrast Function for ICA with the Crosscorrentropy Function

The ICA problem was treated in Chapter 8, and here we use the same basic notation and concepts. The BSS scenario is the same as depicted in Figure 8.2, where for simplicity we are just assuming two sources and two

sensors. For cases where the data have temporal structure, we propose to constrain the Euclidean distance between the output cross-correntropy function of the demixer at different lags to obtain the first correntropy based criterion for BSS (denoted as  $J_1$ ):

$$\min_W J_1 = \sum_{l=-L}^L (v_{y1,y2}(0) - v_{y1,y2}(l))^2, \quad (11.47)$$

where  $y_1(n)$  and  $y_2(n)$  are the temporal source estimates and  $L$  is a user-specified parameter representing the number of lags. Using the simple steepest descent method in batch mode, the update equation for the demixing matrix  $\mathbf{W}$  is:

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \eta \sum_{l=-L}^L (\hat{v}_{y1,y2}(0) - \hat{v}_{y1,y2}(l)) (\nabla_W \hat{v}_{y1,y2}(0) - \nabla_W \hat{v}_{y1,y2}(l)), \quad (11.48)$$

where  $\eta$  is a stepsize parameter and,

$$\begin{aligned} \nabla_W \hat{v}_{y1,y2}(l) &= \frac{1}{N-l} \sum_{n=l+1}^N \kappa(y_1(n) - y_2(n)) \\ &\quad (y_1(n) - y_2(n-l)) \begin{bmatrix} -x_1(n) & -x_2(n) \\ x_1(n-l) & x_2(n-l) \end{bmatrix} \end{aligned} \quad (11.49)$$

for  $l = 0, \pm 1, \dots, \pm L$ .

Although it cannot be directly proved that, for all  $l$  Eq. (11.47) is a sufficient condition for independence, we argue that it still suffices as a criterion for BSS. Indeed, this criterion was first motivated by the joint diagonalization of a set of covariance matrices, as in SOBI [29]. Like the correlation matrix, given  $V[l]$  for  $l = 0, \pm 1, \dots, \pm L$ , we can form the following cross-correntropy matrix:

$$\mathbf{V} = \begin{bmatrix} v(0) & \cdots & v(L) \\ \vdots & \ddots & \vdots \\ v(-L) & \cdots & v(0) \end{bmatrix}.$$

The  $J_1$  criterion is in fact minimizing the Frobenius norm of the matrix  $\mathbf{V} - \mathbf{V}_0$  (with appropriate scaling for different entries), where  $\mathbf{V}_0$  is a matrix with constant entry  $V(0)$ . When the global minimum is achieved, the matrix  $\mathbf{V} - \mathbf{V}_0$ , which contains high-order statistics of the mixtures, becomes a diagonal matrix (in fact a null matrix for  $J_1$ ). In this sense, the above criterion is similar to the method of joint diagonalization of cumulant matrices or covariance matrices for BSS [335], except that now the entries of  $\mathbf{V}$  also contain higher-order statistical information about the sources. For this reason one can expect better performance when compared to SOBI when

the sources are not Gaussian. The criterion  $J_1$  may also be used to separate temporally correlated Gaussian sources because it constrains more than one lag of the mixtures (ICA algorithms will fail in this case). Suppose we have two temporally correlated Gaussian sources and their whitened mixtures, assuming unit variance for both sources:  $y_1[n] = s_1[n] \cos \theta - s_2[n] \sin \theta$ , and  $y_2[n] = s_1[n] \sin \theta + s_2[n] \cos \theta$ .

Using the RKHS notation,  $\Phi(\cdot) = [\varphi_1(\cdot) \varphi_2(\cdot) \dots \varphi_D(\cdot)]^T$  ( $D$  may be infinity), simple calculations yield:

$$\begin{aligned} v_{y_i, y_2}(0) - v_{y_1, y_2}(l) &= \sum_{i=1}^D \mathbb{E}[\varphi_i(s_1(n) \cos \theta - s_1(n) \sin \theta) - \varphi_i(s_1(n) \cos \theta \\ &\quad - s_1(n-l) \sin \theta) \cdot \mathbb{E}[\varphi_i(s_2(n) \cos \theta - s_2(n) \sin \theta) \\ &\quad - \varphi_i(s_2(n-l) \cos \theta - s_2(n) \sin \theta)] \quad (11.50) \end{aligned}$$

Because  $\varphi_i(\cdot)$  are nonlinear functions that involve high-order statistics of the sources, the two expectations for each  $i$  are generally not zero for arbitrary angle  $\theta$ . Finally, because the crosscorrentropy function is constant across the lags for independent sources (Property 11.9),  $J_1$  is especially useful when applied to sources that have a time structure, since it uses both statistical and lag information in the demixing. For i.i.d. sources there is no lag structure in the sources and the criterion can be simplified, as shown next.

### Contrast Function Based on Crosscorrentropy

Instead of looking at  $y_1(n)$  and  $y_2(n)$  as stochastic processes, one can simply drop the time indices and interpret them as two random variables  $y_1$ ,  $y_2$ . Obviously this modification will lose information about the temporal structure of the mixtures. It turns out that for this case, we can obtain yet another simplified criterion for BSS with correntropy, denoted as  $J_2$ , which in this case reduces to an ICA criterion because it only exploits statistical information about the sources at a single lag.

From Property 10.8 we already saw that for independent random variables, cross-correntropy is the inner product of the mean values of the projected data, which was called the cross-information potential (CIP). Neglecting the issue of sufficiency for the time being, we obtain another cost function for ICA based on cross-correntropy (denoted as  $J_2$ ):

$$\min_W J_2 = [v_{y_1, y_2} - CIP_{y_1, y_2}]^2. \quad (11.51)$$

Again, using the steepest descent method in batch mode, the update equation for the demixing matrix  $\mathbf{W}$  is:

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \eta \left( \hat{v}_{y_1, y_2} - \hat{CIP}_{y_1, y_2} \right) \left( \nabla_{\mathbf{W}} \hat{v}_{y_1, y_2} - \nabla_{\mathbf{W}} \hat{CIP}_{y_1, y_2} \right), \quad (11.52)$$

where

$$\nabla_W \hat{v}_{y_1, y_2} = \frac{1}{N-l} \sum_{i=1}^N \kappa(y_1(i) - y_2(i))(y_1(i) - y_2(i)) \begin{bmatrix} -x_1(i) & -x_2(i) \\ x_1(i) & x_2(i) \end{bmatrix} \quad (11.53)$$

$$\nabla_W \hat{CIP}_{y_1, y_2} = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \kappa(y_1(i) - y_2(j))(y_1(i) - y_2(j)) \begin{bmatrix} -x_1(i) & -x_2(i) \\ x_1(j) & x_2(j) \end{bmatrix}. \quad (11.54)$$

From Eq. (11.50), we can see that independence in the input space implies uncorrelatedness in the feature space, with the feature vectors being  $\Phi(y_1)$  and  $\Phi(y_2)$ . Or if we look at the individual nonlinear mapping  $\varphi_i(\cdot)$ , we can see that minimizing the cost function  $J_2$  is in a sense doing nonlinear decorrelation, which was successfully used by Jutten and Herault [172] to solve the first ICA problems. But the difference is, instead of decorrelating one nonlinear term,  $J_2$  is decorrelating the sum of a possibly infinite number of nonlinear terms.

If we use the statistical interpretation for cross-correntropy (as  $p(Y_1 = Y_2)$ ) in the large-sample case, the criterion  $J_2$  is in fact minimizing the following quantity:

$$J = \left[ \int_R p_{y_1 y_2}(u, u) du - \int_R p_{y_1}(u) p_{y_2}(u) du \right]^2. \quad (11.55)$$

This is related to the contrast function based on Euclidean-distance quadratic mutual information of Chapter 2, which takes the functional form:

$$J_{ED-QMI} = \iint_{R^2} [p_{y_1 y_2}(u, v) - p_{y_1}(u)p_{y_2}(v)]^2 dudv. \quad (11.56)$$

For two random variables  $y_1, y_2$  that are independent, we would like  $p_{y_1 y_2}(u, v) = p_{y_1}(u)p_{y_2}(v)$  to be true for each point  $(u, v)$  in the sample space, which is satisfied at the global minimum of Eq. (11.56). However, this is not true for Eq. (11.55), where the only constraint is that the integrations for the joint PDF and product of the marginal PDFs along the line  $u = v$  are (approximately) equal. Obviously, the criterion  $J_2$  is a relaxed version of the contrast function based on quadratic mutual information, with the advantage that it is simpler to implement (just one sum, instead of a double sum). The criterion  $J_2$  also bears some relation to the independence measures based on kernel methods, for example, the constrained covariance proposed by Gretton et al. in [123] as discussed when Property 10.8 was presented.

Because the cost function  $J_2$  does not involve any time structure of the mixtures, it can not be used to separate temporally correlated Gaussian sources. In general we can consider  $J_2$  to be a weaker criterion than  $J_1$ . It is straightforward to generalize the above two criteria to treat more than

two sources by taking the summation of pairwise cost functions defined in Eqs. (11.51) and (11.47), respectively. Ideally at the global minimum of these cost functions, only pairwise independence among the outputs is achieved. In general, it is not guaranteed that the outputs are mutually independent. However, Comon showed that, in the context of linear instantaneous ICA, pairwise independence is equivalent to mutual independence provided that at most one of the independent components is Gaussian and that the underlying densities do not contain delta functions [62]. This means that the new criteria may be suitable for multivariate linear instantaneous ICA problems. But the disadvantage is that the computation cost increases combinatorially with the number of pairs. It may be better to use the extended definition of correntropy for random vectors.

## Demixing Results

In this section we compare the proposed criterion, correntropy ICA with FastICA [155], JADE [46], and SOBI [29]. The performance measure is the signal-to-distortion ratio (SDR), which is given by

$$SDR = -10 \log_{10} \left[ \frac{1}{M(M-1)} \sum_{i=1}^M \left( \frac{\sum_{j=1}^M |B_{ij}|}{\max_j |B_{ij}|} - 1 \right) \right], \quad (11.57)$$

where  $B = WA$ . This measure was initially proposed by Amari, et al [5] and is invariant with respect to permutation and scaling of the rows in  $W$ . It avoids the case of multiple sources in the same channel unlike Eq. (8.50). The argument of the log function is always between 0 and 1 and equal to 0 if and only if there is a perfect match between  $W^{-1}$  and  $A$ , up to permutation and scaling. Intuitively, SDR measures the average proportion of components that are not recovered by the algorithm. In all the following simulations, SDR is averaged over 50 Monte Carlo runs. Each of the Monte Carlo runs uses a different realization of the sources (except in the last example) and a different mixing matrix, whose entries are chosen uniformly between  $-1$  and  $1$ .

The cost function is minimized for prewhitened mixtures using the standard gradient descent in batch mode (although not required for our cost function, we find that it often gave a slightly higher SDR). The parameters to be adapted are the Givens rotation angles as done in Chapter 8 [150]. The kernel size is annealed exponentially during adaptation from  $10\sigma_s$  to  $\sigma_s$ , where  $\sigma_s = 1.06\sigma_x N^{-0.2}$ , according to Silverman's rule, where  $\sigma_x$  is the standard deviation of the data. When it is appropriately used, kernel size annealing avoids local optima and find the global optima more often than the algorithm which uses a static kernel size.

Table 11.4 shows the performance for separation of instantaneous mixtures of speech and music with a fixed sample size of 2000. We note that the

**Table 11.4.** Separation of Instantaneous Mixtures of Speech and Music (From [198])

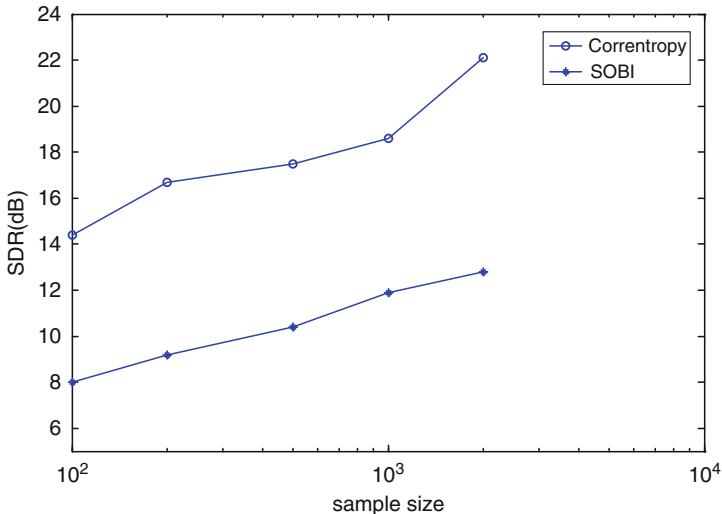
SDR(dB)/method	Correntropy ICA $J_1$	JADE	FastICA	SOBI
Mean	19.7	17.3	15.6	12.2
St. D.	10.4	0.0	7.3	8.2

speech signal has a super-Gaussian distribution with a kurtosis of 0.68 and the music signal has a sub-Gaussian distribution with a kurtosis of  $-0.22$ . In this example,  $J_1$  outperforms all the other algorithms. Nevertheless, the trade-off is that it gives a larger variance in the performance. This means that for some of the Monte Carlo runs,  $J_1$  actually fails to recover the sources. One of the reasons for this is attributed to the local minima on the performance surface.

In general we can say that correntropy ICA seems more suitable to separate mixtures of mixed kurtosis sources (i.e., super-Gaussian and sub-Gaussian) i.i.d. sources. On average, correntropy ICA gives an SDR 4 dB above FastICA and JADE when separating mixtures of uniform and Laplacian sources. It is also efficient for small sample sizes compared with JADE and FastICA, particularly for separation of the two Laplacian sources and mixtures of super- and sub-Gaussian sources. If we linearly regress the SDR by the sample size, the slope for correntropy ICA would be smaller than the other two algorithms. In this sense, we can say that the performance of correntropy ICA in terms of SDR is less sensitive to the sample size.

Figure 11.9 shows the effectiveness of correntropy ICA for separating temporally correlated Gaussian processes that have distinct spectra. We can see on average a 7 dB improvement in SDR from the SOBI algorithm, although the poor performance of SOBI for this particular dataset is a bit surprising.

One of the drawbacks of correntropy ICA is that it is not working as well as the other two methods for mixtures of i.i.d. sources when the distributions of the sources are similar. We speculate that the main reason is the integration of the joint PDF along the line  $u = v$  (45 degree angle in joint space) with a bandwidth associated with the kernel size (see Property 10.4), which makes correntropy not very sensitive to the changes in the rotation angle when the mixtures are close to independent. Thus, the empirical minima in the rotation angle will deviate from the theoretical value to a larger extent than the other methods. Performance is even worse for mixtures of super-Gaussian sources since there are effectively much fewer data to estimate the correntropy along the 45 degree line. This calls for the use of parametric correntropy and a preprocessing step that finds the regions in mixture space where the highest density of samples resides. Although fine tuning of the annealing scheme for each run may increase the performance, this was not done in the experiments and thus the benefits of kernel size annealing may not have been taken full advantage of.



**Fig. 11.9.** Sample size versus SDR for separation of temporally correlated Gaussian sources (from [198]).

## 11.9 Application: CMACE for Automatic Target Recognition

Object recognition involves assigning a class to each observation. Correlation filters have been applied successfully to object recognition by virtue of their linear, shift-invariant, and rotation-invariance through training over the class of images [189]. The application domains range from automatic target detection and recognition (ATR) in synthetic aperture radar (SAR) [210] to biometric identification such as faces, iris, and fingerprint recognition [191]. Object recognition can be performed by cross-correlating an input image with a synthesized template (filter) from previous observations at different poses of the object class in a training set. The correlator output is searched for the peak, which is used to determine whether the object of interest is present.

As we mentioned in Chapter 10, matched filters are the optimal linear filters for signal detection under linear channel and Gaussian noise conditions. However, for image detection, matched spatial filters (MSF) do not fulfill the optimal condition and must be improved. See [190] for an excellent review. Here we perform object recognition using a widely used correlation filter called the minimum average correlation energy filter [209] already described in Section 11.5. The MACE filter tends to produce sharp correlation peaks, however, has been shown to have poor generalization properties; that is, images in the recognition class but not in the training set may not be recognized well. One reason for the poor generalization is that the MACE is still a linear filter, which is optimal only when the underlying statistics are Gaussian. Nonlinear extensions of the MACE have been attempted, including

kernel methods. See [166] for a review. Here we compare the CMACE filter developed in Section 11.3 with the MACE and the kernel MACE on a target recognition application.

This section shows the performance of the proposed correntropy-based nonlinear MACE filter for the SAR image recognition problem in the MSTAR/IU public release dataset [273] already used in Chapter 7. Because the MACE and the CMACE have a constraint at the origin of the output plane, we centered all images and cropped the centered images to the size of  $64 \times 64$  pixels (in practice, with uncentered images, one needs to compute the whole output plane and search for the peak). The selected area contains the target, its shadow, and background clutter. Images lying in the aspect angles of 0 to 179 degrees are used in the simulations. The original SAR image is composed of magnitude and phase information, but here only the magnitude data are used.

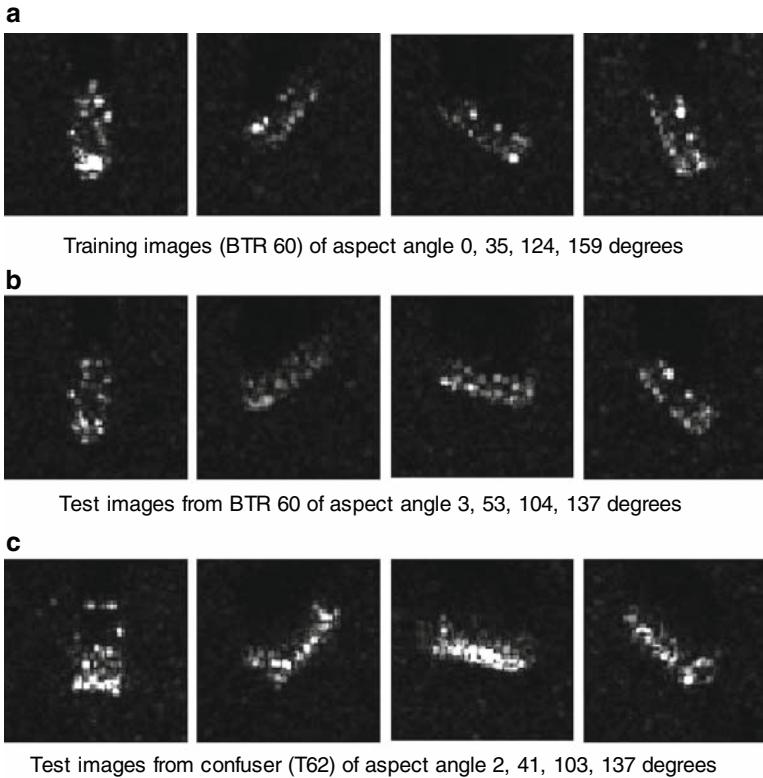
For the first test, a different aspect angle between training and testing data is selected. For the second test, a different depression angle between test and training data is selected, which is a harder problem because the silhouette of the same vehicle may change drastically. In the simulations, the performance is measured by observing the test output peak value and creating the ROC (receiver operating characteristic) curve. The kernel size,  $\sigma$ , is chosen to be 0.1 for the estimation of correntropy in the training images and 0.5 for test output in Eq. (11.35). The value of 0.1 for the kernel size corresponds to the standard deviation of the training data which is consistent with Silverman's rule. Experimentally it was verified that a larger kernel size for testing provided better results.

### Aspect Angle Distortion Case

In the first simulation, we selected the BTR60 (armored personal carrier) as the target (true class) and the T62 (tank) as the confuser (false class). The training and test images are taken at 17 degree depression angles. The goal is to design a filter that will recognize the BTR60 with minimal confusion from the T62. Figure 11.10a shows several examples of training images, used to compose the MACE and the CMACE filters. In order to evaluate the effect of the aspect angle distortion, every third of the 120 images was selected for training ( $N = 40$ ). The remaining images constitute out-of-class exemplars (the difference in aspect angle from the closest training image is 1 or 2 degrees difference).

Figure 11.10b shows test images for the recognition class and Figure 11.10c shows several examples of the confusion vehicle images. Testing is conducted with all 120 exemplar images for each vehicle. We are interested only in the center of the output plane, because the images are already centered. The peak output responses over all exemplars in the test set are shown in Figure 11.11.

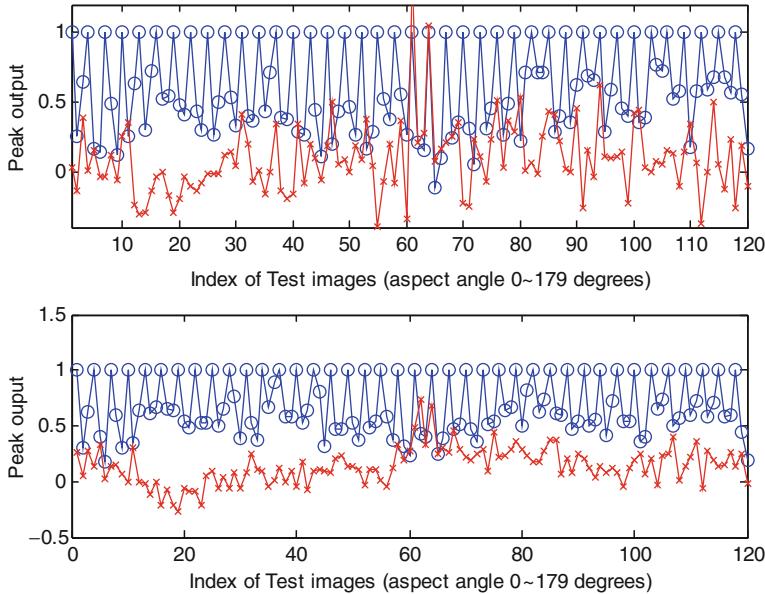
In the simulation, a unity constraint was applied for the MACE as well as the CMACE filter in the training, therefore the desired output peak value



**Fig. 11.10.** Recognition with aspect angle distortion. Sample SAR images ( $64 \times 64$  pixels) of the same vehicle type (BTR60) when the training and the test is taken at different aspect angles, and a confuser (T62) (from [166]).

should be close to one when the test image belongs to the target class and should be close to zero otherwise. We compute results for MACE, CMACE, and the kernel correlation filter (KCF) [338]. For the KCF, prewhitened images are obtained by multiplying the input vector by  $\mathbf{D}^{-0.5}$  in the frequency domain and applying the kernel trick to the prewhitened images to compute the output with the centered covariance. A Gaussian kernel with  $\sigma = 5$  is used for the KCF.

Figure 11.11 (circles) illustrates that recognition is perfect for both the MACE and the CMACE within the training images, as expected. However, it also shows that, in the MACE filter (top), most of the peak output values on the test images are less than 0.5. This indicates that the MACE output generalizes poorly for the images of the same class not used in training, which is one of its known drawbacks. The CMACE (bottom) holds the output of the out-of-class training mostly above 0.5. Figure 11.11 also depicts performance for the confuser test images (crosses). Most of the output values for the MACE (top) are near zero but some are higher than those of target im-



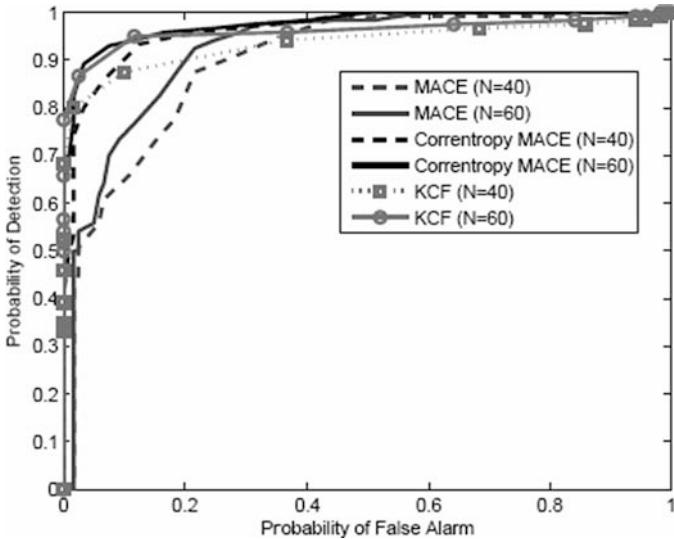
**Fig. 11.11.** Aspect angle distortion case. Peak output responses of testing images for a target chip (circle) and a confuser (cross): (top) MACE, (bottom) CMACE (from [166]).

ages, creating false alarms. On the other hand, for the CMACE (bottom) the rejecting performance for a confuser is better than the MACE. As a result, recognition performance between two vehicles is improved by the CMACE, as best quantified in the ROC curves of Figure 11.12. From the ROC curves in this figure, we can also see that the CMACE outperforms the nonlinear kernel correlation filter in particular for high detection probability. Figure 11.12 shows test set performance for two training runs with a different number of training samples, and in all cases larger training sets improve performance, as expected.

The CMACE performance sensitivity to the kernel size is studied next. Table 11.5 shows the area under the ROC for different kernel sizes. From this table we observe that kernel sizes between 0.01 and 1 provide little change in detectability (kernel size given by Silverman's rule is 0.1). This result corroborates the robustness of performance to changes in the kernel size one order of magnitude up and down from the Silverman's rule value, already mentioned for the matched filter.

### Depression Angle Distortion Case

In the second simulation, we selected the vehicle 2s1 (rocket launcher) as the target and the T62 as the confuser. These two kinds of images have very similar

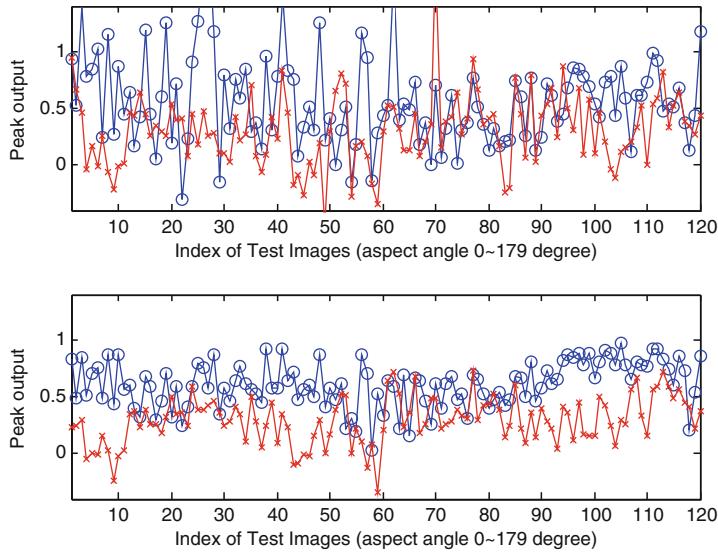


**Fig. 11.12.** Aspect angle distortion case. ROC curves with different numbers of training images. Every filter benefits from a larger training set (from [166]).

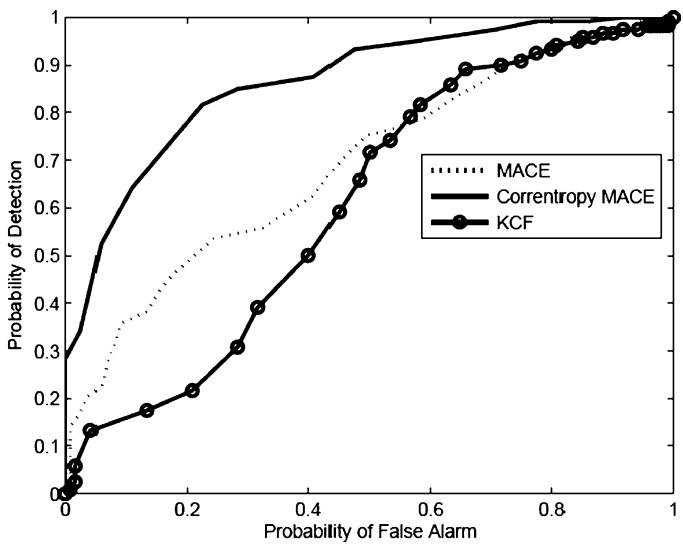
shapes therefore they represent a difficult object recognition case, making it useful for testing purposes. In order to show the effect of the depression angle distortion, the images for the training data have a 13 degree larger depression angle (30 degrees) than the images for the test set (17 degrees). A difference of 13 degrees of depression represents a huge distortion because it substantially increases the shadows and perturbs the object dimensions. In this simulation, we train with all the images (120 images covering 180 degrees of pose) at a 30 degree depression angle and test with all 120 exemplar images at a 17 degree depression angle.

Figure 11.13 (top) shows the correlation output peak value of the MACE and (bottom) the output peak values of the CMACE filter with the target and the confuser test data. We see that the performance of the conventional MACE is very poor in this case, either under- or overshooting the peak value of 1 for the target class, but the CMACE can improve the recognition performance because of its better generalization. Figure 11.14 depicts the ROC curve and summarizes the CMACE advantage over the MACE in this large depression angle distortion case. More interestingly, the KCF performance is closer to the linear MACE, presumably because both use the same input space whitening which is unable to cope with the large distortion.

In practice, the drawback of the proposed CMACE filter is the required storage and its computational complexity. The MACE easily produces the entire correlation output plane by FFT processing. However, computing the whole output plane with the CMACE is a burden. For this reason, we recommend using either the fast Gauss transform or the incomplete Cholesky



**Fig. 11.13.** Depression angle distortion case: peak output responses of testing images for a target chip (circle) and a confuser (cross): (top) MACE, (bottom) CMACE (from [166]).



**Fig. 11.14.** Depression angle distortion case: ROC curves showing the big advantage of the CMACE for this more difficult case (from [166]).

**Table 11.5.** Case A: Comparison of ROC Areas with Different Kernel Sizes (From [166])

Kernel size	0.01	0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
ROC area	0.96	0.96	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.97	0.97

decomposition to save computation time, which results in a computational savings of about 100-fold for  $64 \times 64$  pixel images [166]. More recently we experimented with a compressive sampling scheme that projects the ATR images with a random projection matrix with surprisingly good results [167]. The MACE filter is unable to discriminate the classes from the projected images, but discriminability is preserved to a large extent with the CMACE. This may be due to the inclusion of higher-order statistics in the CMACE generalized correlation.

## 11.10 Conclusion

This chapter presents the autocorrentropy function for random processes and complements the theory presented in Chapter 10 for pairs of random variables. The correntropy function for random processes can be properly called autocorrentropy and it is an enhancement of the autocorrelation function so widely used in time series analysis. However, to define the autocorrentropy function, stationarity in the strict sense is necessary because autocorrentropy is a function of many moments expressed by the kernel.

Perhaps the most interesting feature of the autocorrentropy function is that it combines statistical with temporal information, which are the defining properties of any random process. This feature is due to the fact that the kernel in the correntropy definition expresses higher-order moments of the random variables, yielding a value that is easy to compute but rich in information about the random process.

The other interesting property of the autocorrentropy function is that it is a positive definite bivariate function, and so it defines a RKHS similar in spirit but different in character to the RKHS defined by the autocorrelation function, as proved by Parzen. The difference is that the mapping to the correntropy RKHS is nonlinear, unlike what happens with the autocorrelation function. This results in more powerful representations, but it also introduces difficulties that have not been solved to date. Our results with the CMACE show that when the optimal model order in  $H_v$  is large and when the order of the samples do not matter, the approximation in Eq. (11.36) seems reasonable. Our experience with the same approximation for Wiener filters in prediction (not shown) has not been as successful [248]. Overall, these preliminary results are encouraging which calls for further research in this class of similarity functions.

The chapter also explains properties of correntropy that are related to the time nature of the function. In particular we show that the centered autocorrentropy function accepts a Fourier transform, therefore a correntropy spectral density can be defined, which is still a function of the kernel size. We show that by changing the kernel size a family of spectra representations is obtained, ranging from the PSD to new representations that blend in different degrees second- and higher-order moments of the data. This extra capability has not yet been tested in real problems.

The case studies illustrate four possible advanced signal processing applications: the pitch detection algorithm shows the high performance achieved with autocorrentropy compared with a large set of comparable low-level algorithms. The second example exploits the CIM metric to yield better performance when finding a single sinewave in impulsive noise. However, further work is necessary to fully understand the eigenstructure of the autocorrentropy matrix and apply the method for an arbitrary number of sinewaves. The third example exploits the fact that independent random processes yield a zero correntropy coefficient (unlike the correlation coefficient) and as such a new contrast function for blind source separation based on correntropy is proposed. Currently the crosscorrentropy is calculated along the bisector of the first and third quadrants that may not have sufficient samples to estimate the mixing directions with high precision. Although performance depends on the mixing matrix, parametric correntropy has the potential to make this method very useful as a general-purpose algorithm for BSS if the mixing directions are first estimated to determine the regions in mixture space where autocorrentropy should be estimated. The final example applies autocorrentropy to an image recognition problem, to show that autocorrentropy can be applied to index sets that are not time (in this case the pixels of the image). Here we showed that the performance of the correntropy MACE is superior to that of the MACE in a difficult problem (ATR in SAR), but the computational complexity of CMACE is far higher and will not scale up to large problems. Methods to decrease the computational complexity of the CMACE are required (e.g., FGT and ICD). An alternative tested with surprising results is random subsampling [167].

# A

---

## PDF Estimation Methods and Experimental Evaluation of ITL Descriptors

Deniz Erdogmus and Rodney Morejon

### A.1 Introduction

For optimal processing of the information available in the data at hand we need to obtain reliable estimates of the underlying statistics. Traditionally, parametric Bayesian approaches have been heavily exploited to this goal with very successful results in understanding the data as well as yielding engineering information-processing systems that achieve the desired level of performance. Recently, however, considerable attention has been devoted to semiparametric and nonparametric approaches in signal processing and data analysis, as opposed to the traditional parametric view. In this appendix, several techniques for estimating the data statistics and distributions from samples are summarized. The presentation includes a variety of approaches from both the parametric and nonparametric estimation literatures. The main focus, however, is on nonparametric methods. We also present specific experimental data that demonstrate performance of the information potential and quadratic Renyi's entropy when the experimental parameters of data size, dynamic range, and kernel size are varied. In most real-world learning scenarios the measured signals take continuous values, typically from the set of real numbers and estimation of probability densities and differential information-theoretic quantities is especially difficult. Alternatively the probability mass functions and the corresponding information measures of discrete-valued signals can easily be approximately evaluated by a counting procedure. Due to this reason, we are mostly concerned with estimation of probability densities and differential information measures for continuous random variables throughout this chapter.

### A.2 Probability Density Function Estimation

Density estimation frequently arises when solving learning and signal processing problems. It has been studied extensively in the statistics literature and many useful and strong techniques are at our disposal. Because we are using estimators, it is appropriate to start with basic definitions of estimator quality.

Suppose the data  $\{x_i | i = 1, \dots, N\}$  belongs to a probability density  $p(x)$ . We would like to estimate  $p(x)$  from a finite number of observations by some procedure that has a free parameter  $\sigma$  (e.g., using the histogram  $\sigma$  as the bin width). We will denote the estimator by  $\hat{p}_\sigma(x)$ , and for simplicity the dependence on  $\sigma$  is dropped when no confusion arises. The estimator  $\hat{p}_\sigma(x)$  is called consistent when  $\hat{p}(x) \rightarrow p(x)$ ,  $N \rightarrow \infty$  (i.e., when the *bias* and the *variance* of  $\hat{p}_\sigma(x)$  tend to zero when the number of samples approaches infinity). The bias and variance of the estimator are defined, respectively, as

$$\begin{aligned} & E[\hat{p}_\sigma(x) - p(x)] \\ & \text{Var}[\hat{p}_\sigma(x)] = E[(\hat{p}_\sigma(x) - E[\hat{p}_\sigma(x)])^2]. \end{aligned} \quad (\text{A.1})$$

The mean square error of the estimator is defined as

$$\text{MSE}[\hat{p}_\sigma(x)] = E[\hat{p}_\sigma(x) - p(x)]^2 \quad (\text{A.2})$$

Another quantity normally found in estimation is the *mean integrated square error* (MISE) that is defined as

$$\text{MISE} = E \left[ \int_{-\infty}^{\infty} (\hat{p}_\sigma(x) - p(x))^2 dx \right] = \int_{-\infty}^{\infty} \text{MSE}[\hat{p}_\sigma(x)] dx, \quad (\text{A.3})$$

which can be interpreted as the average global error. Sometimes to obtain close form solutions one makes approximations to the MISE and in this case we call it the asymptotic MISE (AMISE).

The density estimation techniques can be broadly classified as parametric and nonparametric. Here, we discuss both classes of approaches.

## Parametric Density Estimation

Parametric density estimation has been at the core of Bayesian learning and signal processing [38, 79, 180]. These techniques rely on the availability of a parametric family of density functions that can accurately describe the observed data. Once a suitable family of densities is selected, the parameters of the family can be optimized according to the available samples, hence the name *parametric density estimation*. The two most commonly reported approaches in the literature are maximum likelihood estimation and Bayesian estimation. The problem of parametric density estimation from available data can be summarized as follows. Given the set  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  of independent and identically distributed (i.i.d.) random vector samples and a family of density functions  $p(\mathbf{x}; \boldsymbol{\theta})$  from which these samples are drawn, determine the parameter  $\boldsymbol{\theta}_*$  such that the selected distribution  $p(\mathbf{x}; \boldsymbol{\theta}_*)$  and the observed samples  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  exhibit the best fit with each other.

*Maximum Likelihood Parameter Estimation.* According to the maximum likelihood (ML) principle for estimating the parameters of an underlying density function, the available samples are generated by a distribution whose

parameters are fixed, but are unknown to us. Consequently, the conditional distribution  $p(\mathbf{x}|\theta_*)$  is assumed to be the underlying density for the observed i.i.d. samples  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ . For an arbitrary value  $\theta$  of the parameters, the probability of observing a particular sample  $\mathbf{x}_k$  is  $p(\mathbf{x}_k|\theta)$ . The observed samples are independent, therefore the joint probability distribution of the complete dataset for the given parameter vector  $\theta$  is the product of these individual conditional probability densities.

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N|\theta) = \prod_{k=1}^N p(\mathbf{x}_k|\theta), \quad (\text{A.4})$$

This quantity is referred to as the likelihood of the data for the given parameter vector  $\theta$ . The ML estimate for the underlying density is obtained by maximizing this likelihood function with respect to  $\theta$ . In general, the logarithm of the likelihood function is considered instead, because the product simplifies to a summation. Then, the ML estimate of the best density parameter corresponds to the solution of the following maximization problem.

$$\hat{\theta}_{ML} = \arg \max_{\theta} \sum_{k=1}^N \log p(\mathbf{x}_k|\theta), \quad (\text{A.5})$$

where the optimality measure is referred to as the log-likelihood function and is typically denoted by  $L(\theta)$ .

The ML principle is known to correspond to a minimum Kullback–Leibler (KL) divergence estimate for the underlying density. In order to observe this relationship, we consider the expected value of the log-likelihood function defined in Eq. (A.5) with respect to the true distribution  $q(\mathbf{x})$  of the random vector  $\mathbf{X}$  with samples  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ , which does not necessarily lie within the selected family of distributions  $p(\mathbf{x}; \theta)$  (also denoted by  $p_\theta$  in the following derivation). Consider the following minimum KL problem for determining the optimal density parameters.

$$\begin{aligned} \hat{\theta}_{KLD} &= \arg \min_{\theta} D_{KL}(q; p_\theta) \\ &= \arg \min_{\theta} \int q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p(\mathbf{x}; \theta)} d\mathbf{x} \\ &= \arg \min_{\theta} \int q(\mathbf{x}) \log q(\mathbf{x}) d\mathbf{x} - \int q(\mathbf{x}) \log p(\mathbf{x}; \theta) d\mathbf{x} \\ &= \arg \min_{\theta} \int q(\mathbf{x}) \log q(\mathbf{x}) d\mathbf{x} - \int q(\mathbf{x}) \log p(\mathbf{x}; \theta) d\mathbf{x} \\ &= \arg \min_{\theta} -H_S(\mathbf{X}) - E_{\mathbf{X}}[\log p(\mathbf{X}; \theta)]. \end{aligned} \quad (\text{A.6})$$

In the last line of Eq. (A.6),  $H_S(\mathbf{X})$  denotes the Shannon entropy of the true underlying distribution  $q(\mathbf{x})$ , which is a constant with respect to the

optimization variable  $\theta$ . Therefore, this term can be dropped from the KL optimization criterion. In addition, the minimization problem can be replaced by a maximization problem if the negative sign in the second term is omitted. Finally, we can approximate the expectation of  $\log p(\mathbf{X}; \theta)$  with respect to  $\mathbf{X}$  using the sample mean approximation. With these assumptions, Eq. (A.6) becomes

$$\begin{aligned}\hat{\theta}_{KL} &= \arg \min_{\theta} D_{KL}(q || p_{\theta}) \\ &= \arg \max_{\theta} \frac{1}{N} \sum_{k=1}^N \log p(\mathbf{x}_k; \theta) \\ &= \arg \max_{\theta} \frac{1}{N} L(\theta) \\ &= \hat{\theta}_{ML}.\end{aligned}\quad (\text{A.7})$$

Recall from the Pythagorean theorem of relative entropy that for  $p_{\theta}$ ,  $p_{\theta^*}$ , and  $q$  we have  $D_{KL}(p_{\theta} || q) \geq D_{KL}(p_{\theta} || p_{\theta^*}) + D_{KL}(p_{\theta^*} || q)$ . Provided that the conditions in the theorem are satisfied by the family  $p_{\theta}$  and that minimizing  $D_{KL}(p_{\theta} || q)$  and  $D_{KL}(q || p_{\theta})$  yield the same  $p_{\theta^*}$ , then we can expect the ML estimate  $\hat{\theta}_{ML}$  to approximate  $\theta_*$ . Thus the corresponding ML density estimate  $p_{\theta^*}$  becomes the optimal projection of the true density  $q$  in the subset of distributions spanned by the selected family  $p_{\theta}$ .

*Bayesian Parameter Estimation.* In contrast to the ML parameter estimation principle, here we treat the parameter vector  $\theta$  as a random variable. The available data, in Bayesian parameter estimation, are utilized to convert a prior distribution knowledge on this random variable to a posterior probability distribution. Consequently, this method is also referred to as maximum a posteriori (MAP) parameter estimation principle.

Suppose that the available i.i.d. samples  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  are drawn from a distribution  $p(\mathbf{x}|\theta_*)$ . We assume that we know the following: the parametric family of distributions  $p(\mathbf{x}|\theta)$  and the a priori distribution of the parameter vector  $p(\theta)$ . For conditional distributions, we know that the following identity holds:  $p(\mathbf{x}_1, \dots, \mathbf{x}_N|\theta)p(\theta) = p(\theta|\mathbf{x}_1, \dots, \mathbf{x}_N)p(\mathbf{x}_1, \dots, \mathbf{x}_N)$ . From this, the a posteriori probability distribution of the parameter vector given the data are found to be

$$p(\theta|\mathbf{x}_1, \dots, \mathbf{x}_N) = \frac{p(\mathbf{x}_1, \dots, \mathbf{x}_N|\theta)p(\theta)}{p(\mathbf{x}_1, \dots, \mathbf{x}_N)}.\quad (\text{A.8})$$

The MAP estimate for the parameter of the distribution underlying the data are then obtained by maximizing the a posteriori distribution in Eq. (A.8) with respect to  $\theta$ . The denominator on the right-hand side is independent of the estimated parameter  $\theta$ , so it can be dropped from the optimization criterion. As in the ML estimation procedure, because the samples are assumed to

be i.i.d. with distributions  $p(\mathbf{x}_k|\theta)$ , taking the log of the criterion yields the following problem that gives the MAP estimate,

$$\hat{\theta}_{MAP} = \arg \max_{\theta} \left( \log p(\theta) + \sum_{k=1}^N \log p(\mathbf{x}_k|\theta) \right). \quad (\text{A.9})$$

We note that this is similar to the ML problem defined in Eq. (A.5), the difference being the introduction of some a priori information about the value of the underlying parameter in the form of a probability distribution  $p(\theta)$ . Notice that if  $p(\theta)$  is assumed to be uniform over some set of possible values for the parameter, MAP estimation reduces to ML estimation over this set of values. In addition, if we denote the optimality criterion in Eq. (A.9) by  $M(\theta)$  and equivalently consider the maximization of  $M(\theta)/N$ , as the number of samples  $N \rightarrow \infty$ , the importance of the a priori distribution (i.e.,  $(1/N) \log p(\theta)$ ) will decay to zero. Thus, we conclude that asymptotically  $\hat{\theta}_{MAP} \rightarrow \hat{\theta}_{ML}$ . Hence we expect it to approach the minimizer of the KL between the estimated distribution and the true underlying distribution, even if the true distribution  $q(\mathbf{x})$  is not in the parametric family  $p(\mathbf{x}|\theta)$ , as discussed above.

## Semiparametric Density Estimation

In parametric density estimation, the requirement to assume a family of densities denoted by  $p(\mathbf{x}|\theta)$  is quite restrictive. The assumed family of distributions will often have few parameters to optimize: for example, if a Gaussian distribution is assumed, only the mean and covariance must be determined. The assumed distribution is typically one of the well-established families such as exponential, gamma, beta, Rician, Rayleigh, and so on. The performance of the following steps in the information-processing procedure relies strongly on the success of the selected family of distributions in representing the data distribution adequately.

*Mixture Density Models.* Especially in machine learning and adaptive signal processing, where the data distribution might be constantly changing due to the weights adapting in time, the selected family should be sufficiently complicated to accommodate a wide variety of distributions. Under these circumstances, one commonly resorts to the semiparametric density estimation methods. One commonly used semiparametric approach is the mixture density paradigm where the data distribution is assumed to be within a family of mixture distributions such as the mixture of Gaussians. The mixture density is composed of a linear combination of parametric density models, where the number of models and their linear combination coefficients can be separately optimized. If we denote a Gaussian distribution with mean  $\mu$  and covariance  $\Sigma$  by  $G(\mathbf{x}; \mu, \Sigma)$ , then a mixture of  $m$  Gaussians is given by

$$p(\mathbf{x}; \{\alpha_i, \mu_i, \Sigma_i\}) = \sum_{i=1}^m \alpha_i G(\mathbf{x}; \mu_i, \Sigma_i), \quad (\text{A.10})$$

where the mixture coefficients  $\alpha_i$  must be positive and add up to one; that is,  $\alpha_1 + \dots + \alpha_m = 1$ . A natural interpretation of this mixture model comes from the total probability theorem. We understand from Eq. (A.10) that the data are generated by a random process that first selects which Gaussian model  $G(\mathbf{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$  is to be used, with each model having probability  $\alpha_i$ ; then a random sample is drawn from the selected Gaussian distribution.

In general we need not use all Gaussian distributions in the mixture model. In fact the mixture can be composed of various distributions as long as each component is a valid probability distribution itself. The mixture of Gaussians is typically preferred for convenience (because only the mean and the covariance of each Gaussian mode need to be optimized using ML or MAP techniques). The added benefit of using Gaussian mixtures is that it is known to be able to approximate any smooth distribution asymptotically as the number of models is increased. This approximation capability is a direct consequence of the universal approximation capability of radial basis function (RBF) networks [141].

In semiparametric density estimation, the parameters are still optimized using the ML principle. An efficient algorithm that implements the ML procedure is called the expectation–maximization (EM) algorithm [70], but it is not reviewed here. Another method that can be considered semiparametric is Jaynes’ maximum entropy estimation principle, in as much as it is based on a truncated moment expansion of the PDF [161].

*Series Expansion Approximations.* The maximum entropy estimate using moment constraints mentioned above can be considered as a truncated Taylor series approximation of an exponential distribution. In fact, a number of such truncated series approximations for distributions exist. We review below Legendre and Gram–Charlier series expansions of PDFs.

*Legendre series expansion of a distribution:* Consider the Legendre polynomials given recursively by

$$P_k(x) = \begin{cases} 1 & k = 0 \\ x & k = 1 \\ \frac{1}{k} [(2k-1)xP_{k-1}(x) - (k-1)P_{k-2}(x)] & k \geq 2 \end{cases}. \quad (\text{A.11})$$

Any PDF  $f(x)$  can be expressed as an infinitely weighted sum of Legendre polynomials as shown below.

$$f(x) = \sum_{k=0}^{\infty} \left( k + \frac{1}{2} \right) E[P_k(X)] P_k(x). \quad (\text{A.12})$$

*Gram–Charlier series expansion of a distribution:* The characteristic function of a distribution  $f(x)$ , denoted by  $\Phi_f(\omega)$ , can be expressed in terms of the characteristic function  $\Phi_p(\omega)$  of an arbitrary reference PDF  $p(x)$  as

$$\Phi_f(\omega) = \exp \left( \sum_{k=1}^{\infty} (c_{f,k} - c_{p,k}) \frac{(j\omega)^k}{k!} \right) \Phi_p(\omega), \quad (\text{A.13})$$

where  $c_{f,k}$  and  $c_{p,k}$  are the  $k$ th cumulants of  $f(x)$  and  $p(x)$ , respectively, and  $j = \sqrt{-1}$ . The cumulants can be expressed in terms of the moments of a distribution using the cumulant generating function.

For the special case of a zero-mean and unit-variance Gaussian distribution as the reference PDF (denoted by  $G(x)$  in the following), expanding the series and collecting the same order derivatives of the PDF together, an equivalent expansion can be obtained in terms of the Hermite polynomials. This yields,

$$f(x) = G(x) \cdot \sum_{k=0}^{\infty} C_k H_k(x), \quad (\text{A.14})$$

where the coefficients of the expansion are simply  $C_k = E[H_k(X)]/k!$  and the Hermite polynomials are obtained by the recursion  $H_k(x) = xH_{k-1}(x) - (k-1)H_{k-2}(x)$ , with  $H_1(x) = x$  and  $H_0(x) = 1$ . Alternatively, the Hermite polynomials corresponding to the selected reference PDF  $G(x)$  can be obtained from the following Taylor series expansion:

$$\exp(tx - t^2/2) = \sum_{k=0}^{\infty} \frac{t^k}{k!} H_k(x). \quad (\text{A.15})$$

*Expansion of a distribution using complete bases:* In general, all distributions in the PDF space can be approximated by using a truncated subset of bases. Given a complete basis set composed of infinite linearly independent functions  $\{b_1(x), b_2(x), \dots\}$ , it is possible to express an arbitrary PDF  $f(x)$  belonging to this space by a linear combination. Specifically, if the basis functions are selected to be orthonormal (such as Laguerre functions), the determination of the coefficients becomes trivial. Denoting the orthonormal bases obtained using the Laguerre functions by  $\{\varphi_1(x), \varphi_2(x), \dots\}$ , the PDF  $f(x)$  is given by

$$f(x) = \sum_{k=1}^{\infty} E[\varphi_k(X)] \varphi_k(x), \quad (\text{A.16})$$

where  $E[\varphi_k(X)]$  corresponds to the inner product of the PDF with the  $k$ th basis function in this Hilbert space. This inner product is defined as

$$\langle f(x), \varphi_k(x) \rangle = \int f(x) \varphi_k(x) dx = E[\varphi_k(X)]. \quad (\text{A.17})$$

The polynomial expansions are simply special cases of this approach. In practice, the expectations that determine the expansion coefficients can be approximated by using sample mean estimates.

A drawback of expansions for density estimation is that the truncation might result in estimates that do not satisfy the two basic properties of a PDF: nonnegativity and integrating to unity. Although from a density estimation perspective this is undesirable, in some information-theoretic signal processing approaches, the cost function estimates resulting from such density approximations might perform satisfactorily.

## Nonparametric Density Estimation

Nonlinear adaptive processing of real-world signals and data often requires density models that can cope with a wide variety of signal distribution possibilities. Parametric models are quite restricted in their representation capability and semiparametric models offer some relaxation of these restrictions. However, in many cases, one still needs to make assumptions that limit the representation capability of semiparametric models. For example, in Gaussian mixture models we need to specify a priori the number of Gaussian modes that will be utilized, or in the maximum entropy approach with moment constraints we need to specify the order moments that will be considered. Often, these decisions have to be done blindly, in other words, without any knowledge of the signals with which we will be dealing.

From this point of view, nonparametric density estimation techniques offer the most freedom for representing signal distributions based purely on the observed samples of data. A variety of nonparametric density estimation methods is available in the statistics literature; however, we focus on only a few of these in this section. Most important, we discuss Parzen windowing, which is also referred to as *kernel density estimation*, as estimators of information-theoretic quantities based on this density estimator are central to the adaptive signal processing and learning algorithms that we discuss.

*Histogram-based Density Estimation.* The simplest nonparametric density estimation method is the histogram. This estimate is motivated by the fact that the probability density function is the limiting value of the following expression:

$$p(x) = \lim_{\Delta \rightarrow 0^+} \frac{P(x - \Delta < X \leq x)}{\Delta}.$$

We only have a finite number of samples in practice, thus taking the limit of the bin size  $\Delta$ , to zero only creates a density estimate that consists of Dirac- $\delta$  spikes located at the samples. This undesirable density estimate can be *smoothed* by letting the bin size stay at a finite value. For a fixed bin size  $\Delta > 0$ , we form the non-overlapping bins centered at  $\{\dots, -k\Delta, \dots, \Delta, 0, \Delta, \dots, k\Delta, \dots\}$ . Then, we count the number of samples that fall in each bin to obtain a sequence called bin counts:  $\{\dots, N_{-k}, \dots, N_{-1}, N_0, N_1, \dots, N_k, \dots\}$ . Clearly, the bin counts must add up to  $N$ , the total number of samples. The histogram estimate for the probability density function is then obtained by a combination of uniform distributions in each bin, where the rectangular area is set equal to the frequency of the samples in that bin:  $f_k = N_k/N$ . Consequently, the height of each uniform bin is  $p(x) = f_k/\Delta$  if  $x \in [k - 1/2, k + 1/2]$ . In practice, as the number of samples increases, the bin size can be decreased slowly. The bin size is key to the trade-off between estimation bias and estimation variance.

The idea of histograms can easily be generalized to multivariate density estimation by expanding the concept of single-dimensional bins to multidimensional

bins with fixed volume. The counting procedure within each bin is repeated and the probability density in each bin is obtained by dividing the frequency by the volume of the bin.

*Nearest Neighbor Density Estimation.* A problem with histograms is the possibility of having many bins that have zero samples in them. A possible modification to this type of density estimation would be to allow the bin size to vary while keeping the number of samples in each bin constant. To achieve this, we select a constant integer  $K$  typically much smaller than the available number of samples, denoted by  $N$ . The probability density function can now be estimated using the sample–frequency approximation given above:  $p(x) = f/V(x)$ . In this estimate the frequency is constant at  $f = K/N$  and the bin size (or the volume in multivariate cases) is determined by the distance to the  $K$ th nearest sample of  $x$  in the sample set. In the univariate case,  $V(x) = 2d_K(x)$ , where  $d_K(x)$  is the distance from  $x$  to its  $K$ th nearest neighbor. In the multivariate case,  $V(\mathbf{x})$  would be the volume of the smallest hypercube (or hypersphere) centered at  $\mathbf{x}$  that contains the  $K$  nearest samples. Consequently, this method is referred to as  $K$ –nearest neighbor density estimation. A typical and reasonable choice is  $K = \sqrt{N}$ .

This approach will yield continuous density estimates but its gradient with respect to the sample points will have discontinuities. This is an undesirable property from an adaptation and learning perspective, because learning algorithms typically rely on gradient–based incremental weight vector updates.

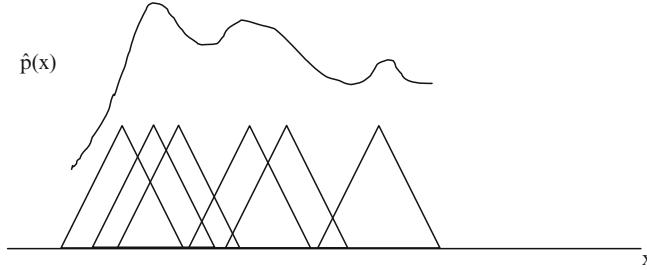
*Kernel Density Estimates.* A nonparametric density estimation approach that yields both continuous and smooth estimates is Parzen windowing [241]. Its basic principle stems from the fixed bin-width approach in histogram density estimates, but the nonoverlapping bins restriction is replaced by a sliding bin method. Assuming rectangular window functions (called kernels from hereon), the density estimate at any point would simply be the number of samples falling within the rectangular window (as in histogram estimates), where the window is symmetric and centered at the point of interest. We follow here closely the exposition of Hardle [133]. A kernel is a density function

$$\int_{-\infty}^{\infty} \kappa(u) du = 1, \quad \text{and} \quad \kappa(u) \geq 0 \quad (\text{A.18})$$

that is symmetric  $\int u\kappa(u) du = 0$ . Table A.1 presents a short list of well-known kernel functions.

**Table A.1.** Examples of Kernel Functions

Uniform	$1/2  u  \leq 1$
Triangle	$(1 -  u )  u  \leq 1$
Epanechnikov	$3/4(1 - u^2)  u  \leq 1$
Quartic	$15/16(1 - u^2)^2  u  \leq 1$
Triweight	$35/32(1 - u^2)^3  u  \leq 1$
Cosinus	$\pi/4 \cos(\pi/2u),  u  \leq 1$



**Fig. A.1.** Estimation of PDF using triangular kernels.

Assuming a hypercubic kernel denoted by  $\kappa_{\mathbf{h}}(\mathbf{x})$ , where the kernel size parameter vector  $\mathbf{h}$  controls the width of the window, the density estimate for a sample set  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  is

$$\hat{p}_\sigma(x) = \frac{1}{N\sigma} \sum_{n=1}^N \kappa\left(\frac{x - x_i}{\sigma}\right). \quad (\text{A.19})$$

The kernel function, in this case, simply evaluates as  $1/V_{\mathbf{h}}$  if  $\mathbf{x}_i$  is inside the hypercube centered at  $\mathbf{x}$  with volume  $V_{\mathbf{h}}$ . That is, it implements the counting operation required in the histogram construction procedure. The normalization by the volume of the hypercube is necessary to obtain a density estimate that integrates to unity over the set of possible values of  $\mathbf{x}$ . Basically the kernel density estimation is a convolution of the true density with the histogram. Figure A.1 illustrates the procedure.

In the general framework of Parzen windowing, the rectangular kernels can be replaced by smoother kernel functions (such as a Gaussian distribution function). In general, the kernel function in Eq. (A.19) can be selected to be any continuous and smooth (differentiable at all orders) feasible zero-mean probability density function itself. Typically, the kernel function is unimodal and symmetric around zero (its mean value) implying that the probability density contribution due to any given sample is maximum at the sample location and decays as one moves away from the sample. For convenience, the Gaussian distribution is preferred due to its many properties. In the case of multidimensional density estimation using Gaussian kernels, the kernel size parameter is, in general, controlled by the full covariance matrix  $\Sigma$  of the Gaussian distribution:

$$G_\Sigma(s) = \frac{e^{-(s^T \Sigma^{-1} s)/2}}{(2\pi)^{n/2} |\Sigma|^{1/2}}. \quad (\text{A.20})$$

Traditionally, a circular Gaussian kernel with  $\Sigma = \sigma^2 \mathbf{I}$  is preferred [79]. Another simple but more general alternative assumes a diagonal  $\Sigma$ , thus a multivariate Gaussian kernel that is separable into a product of univariate Gaussian kernels [86].

Parzen windowing provides a suitable density estimate for the purpose of this book. Specifically for gradient-based learning and adaptation based on information-theoretic criteria, we need continuous and differentiable estimates of relevant information-theoretic quantities, which in turn rely on the underlying density estimation procedure. Therefore, it is important to investigate the convergence properties of Parzen windowing at this point. Suppose that the sample set  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  is *i.i.d.* and drawn from the true underlying density function  $p(\mathbf{x})$ . First, consider the expected value of the Parzen window estimate using a suitable kernel function  $\kappa_\sigma(\mathbf{x})$ :

$$E[\hat{p}(\mathbf{x})] = \frac{1}{N} \sum_{i=1}^N E[\kappa_\sigma(\mathbf{x} - \mathbf{x}_i)] = \frac{1}{N} \sum_{i=1}^N \int p(s) \kappa_\sigma(\mathbf{x}_i - s) ds = p(\mathbf{x}) * \kappa_\sigma(\mathbf{x}), \quad (\text{A.21})$$

which becomes the convolution for continuous  $\mathbf{x}$ . In the case of a finite number of samples, the Parzen window estimate for the PDF yields a biased estimate in general. Because, on average, the estimated distribution is equal to the convolution of the true distribution  $p(\mathbf{x})$  and the selected kernel function  $\kappa_\sigma(\mathbf{x})$ , if the kernel is a symmetric unimodal density as suggested, for very large kernel sizes the estimated PDF is highly biased towards the kernel function itself. On the other hand, as the kernel size approaches zero, the kernel approaches a Dirac-delta, thus the bias in the average estimate vanishes. More formally we can compute the bias as

$$\begin{aligned} E[\hat{p}_\sigma(x)] &= \int \kappa(s) \{p(x) + \sigma s p'(x) + \frac{\sigma^2 s^2}{2} p''(x) + O(\sigma^2)\} ds \\ &= p(x) + \frac{\sigma^2}{2} p''(x) \int s^2 \kappa(s) ds + O(\sigma^2) \\ &\approx p(x) + \frac{\sigma^2}{2} p''(x) \mu_2(\kappa) \quad \text{for } \sigma \rightarrow 0, \end{aligned} \quad (\text{A.22})$$

where  $\mu_2(\kappa) = \int s^2 \kappa(s) ds$ . Therefore,

$$\text{Bias}\{\hat{p}_\sigma(x)\} = E[\hat{p}_\sigma(x)] - p(x) \approx \frac{\sigma^2}{2} p''(x) \mu_2(\kappa). \quad (\text{A.23})$$

However, in practice, the kernel size cannot be reduced arbitrarily as it will increase the estimation variance towards infinity. To see this, consider the variance of the density estimate.

$$\begin{aligned} \text{Var}[\hat{p}(\mathbf{x})] &= \frac{1}{N^2} \sum_{i=1}^N \text{Var}[\kappa_\sigma(\mathbf{x} - \mathbf{x}_i)] = \frac{1}{N} \text{Var}[\kappa_\sigma(\mathbf{x} - s)] \\ &= \frac{1}{N} \left[ \int p(s) \kappa_\sigma^2(\mathbf{x} - s) ds - (p(\mathbf{x}) * \kappa_\sigma(\mathbf{x}))^2 \right] \\ &= \frac{1}{N} [p(\mathbf{x}) * \kappa_\sigma^2(\mathbf{x}) - (p(\mathbf{x}) * \kappa_\sigma(\mathbf{x}))^2]. \end{aligned} \quad (\text{A.24})$$

Consequently, in practice, the kernel size parameters must be selected to optimize this bias–variance trade-off. If we want to estimate the variance we have to evaluate

$$\text{Var}[\hat{p}(x)] = \text{Var} \left[ \frac{1}{N\sigma} \sum_{i=1}^N \kappa \left( \frac{x - x_i}{\sigma} \right) \right] \approx \frac{1}{N\sigma} \|\kappa\|_2^2 p(x) \quad (\text{A.25})$$

for  $N\sigma \rightarrow \infty$  and where we define  $\|\kappa\|_2^2 = \int \kappa^2(s) ds$ .

If we are interested in the MSE of kernel estimates we have to evaluate

$$\text{MSE}[\hat{p}(x)] = \mathbb{E}[(\hat{p}(x) - p(x))^2] \approx \frac{\sigma^4}{4} p''(x)^2 \mu_2(\kappa)^2 + \frac{1}{N\sigma} \|\kappa\|_2^2 p(x) \quad (\text{A.26})$$

for  $\sigma \rightarrow 0$  and  $N\sigma \rightarrow \infty$ , which shows a clear trade-off between variance and bias for the estimator because the bias increases with the kernel size and the variance decreases with it.

*Selecting the kernel size:* For optimal density estimation, careful selection of the kernel function is critical. As demonstrated above, for a given kernel function, large kernel sizes will result in high-density estimation bias, whereas small kernel sizes will lead to high estimation variance. The MISE can give us a way to optimally select the kernel size. Indeed for any kernel, we have

$$\begin{aligned} \text{MISE}[\hat{p}_\sigma(x)] &= \int_{-\infty}^{\infty} \text{MSE}[\hat{p}_\sigma(x)] dx \\ &= \frac{\sigma^4}{4} p''(x)^2 \mu_2(\kappa)^2 + \frac{1}{N\sigma} \|\kappa\|_2^2 + O\left(\frac{1}{N\sigma}\right) + O(\sigma^4) \end{aligned} \quad (\text{A.27})$$

and so for  $\sigma$  going to zero and  $N\sigma$  to infinity, the AMISE is

$$\text{AMISE}[\hat{p}_\sigma(x)] = \frac{\sigma^4}{4} \|p''(x)\|_2^2 \mu_2(\kappa)^2 + \frac{1}{N\sigma} \|\kappa\|_2^2, \quad (\text{A.28})$$

where

$$\sigma_{opt} = \left( \frac{\|\kappa\|_2^2}{N \|p''(x)\|_2^2 \mu_2(\kappa)^2} \right)^{1/5} \propto N^{-1/5}.$$

Therefore, the minimum value depends upon the data and the kernel but we see that optimal value is also related to the number of samples. We can expect that the  $\text{MISE}(\sigma_{opt}) \sim n^{-4/5}$ . However, notice that normally the PDF is not known so there are significant challenges in finding the optimal kernel size for a given application. There are two basic approaches: approximate the second derivative or use cross-validation techniques.

Silverman's rule [300], perhaps the most widely used rule of thumb for kernel density estimation, belongs to the first method and approximates the scalar data by a Gaussian of unknown variance to obtain

$$\|p''\|_2^2 = \sigma^{-5} \int G''(s)^2 ds \approx 0.212\sigma^{-5} \quad (\text{A.29})$$

which yields for a Gaussian kernel

$$\hat{\sigma}_S = 1.06 \hat{\sigma} N^{-1/5} \quad (\text{A.30})$$

for one-dimensional data.

The Park and Marron [237] estimator approximates the second derivative of the PDF by

$$\hat{p}_{\tilde{\sigma}}''(x) = \frac{1}{N\tilde{\sigma}^3} \sum_{i=1}^N \kappa''\left(\frac{x - x_i}{\tilde{\sigma}}\right)$$

which yields

$$\hat{\sigma}_{PM} = \left( \frac{\|\kappa\|_2^2}{N \|\hat{p}_{\tilde{\sigma}}''(x)\|_2^2 \mu_2(\kappa)^2} \right)^{1/5}. \quad (\text{A.31})$$

Instead of estimating the derivative, one can estimate the MISE

$$MISE(\hat{p}_{\sigma}) = \int (\hat{p}_{\sigma}(x) - p(x))^2 dx = \|\hat{p}_{\sigma}\|_2^2 - 2 \int \hat{p}_{\sigma}(x)p(x)dx + \|p\|_2^2. \quad (\text{A.32})$$

Note that only the two first terms depend upon the estimator. Substituting the kernels yields

$$\begin{aligned} J(\sigma) &= \|\hat{p}_{\sigma}\|_2^2 - 2 \int \hat{p}_{\sigma}(x)p(x)dx = \frac{1}{N^2\sigma} \sum_{i=1}^N \sum_{j=1}^N \kappa \otimes \kappa \left( \frac{x_i - x_j}{\sigma} \right) \\ &\quad - \frac{1}{N(N-1)\sigma} \sum_{i=1}^N \sum_{j=1}^N \kappa \left( \frac{x_i - x_j}{\sigma} \right), \end{aligned} \quad (\text{A.33})$$

where  $\otimes$  means convolution. Therefore, we should minimize this quantity w.r.t.  $\sigma$ ; that is,  $\hat{\sigma}_J = \arg \min_{\sigma} J(\sigma)$ .

The bottom line is that there is no single best method to choose the kernel size, so one needs to be careful and establish best procedures to select  $\sigma$  and the kernel. For the estimation of smooth densities on  $R^n$ , it is known that asymptotically, among nonnegative kernels, the  $L_1$ -optimal and  $L_2$ -optimal<sup>1</sup> kernel function is the Epanechnikov kernel, which is given by

$$K_{\sigma}(\mathbf{x}) = \frac{c}{\sigma} \max \left[ \left( 1 - \|\mathbf{x}/\sigma\|_2^2 \right)^n, 0 \right]. \quad (\text{A.34})$$

---

<sup>1</sup> Asymptotic  $L_q$ -optimality is defined by minimizing the cost function  $\int |\hat{p}(x) - p(x)|^q dx$  between the estimated and the true underlying probability density functions fastest as number of samples increase unboundedly.

Specifically, provided that the true underlying distribution is sufficiently smooth (i.e.,  $p(x)$  and  $p'(x)$  absolutely continuous) and not long-tailed (i.e.,  $\int p^{1/2}(x)dx < \infty$ ), the convergence of any bounded and symmetric kernel is uniform with the  $L_1$ -error being bounded from above by  $CN^{-2/5}$ . The constant  $C$  of this upper bound depends on the kernel function as well as the true distribution, and, it is smallest for the Epanechnikov kernel [71].<sup>2</sup> Consequently, although the kernel function in Eq. (A.23) is optimal in terms of convergence speed for a wide class of distributions, it results in density estimates that are not continuously differentiable at places.

A closer look at this problem shows that the value of  $C$  is really not very different among different kernels (no more than 1% difference in efficiency w.r.t. the Epanechnikov kernel) [85]. Because the Gaussian kernel of Eq. (A.20) is bounded and symmetric, its convergence rate is also  $O(N^{-2/5})$ . Besides, it exhibits additional desirable properties such as smoothness, as well as invariance under convolution (in other words, the convolution of two Gaussian functions is also Gaussian).<sup>3</sup> The statistics literature on kernel density estimates mainly deals with the  $L_q$ -error norms whereas in signal processing maximum likelihood approaches are often preferred.

*ML kernel size estimation.* Especially in the context of this text, ML approaches become more important as they relate closely to the information-theoretic techniques that we seek to incorporate into signal processing. Next, we present the ML approach for kernel size optimization. Although, in practice, we mostly utilize the Gaussian kernel, for the sake of generality, we formulate the problem in terms of an arbitrary symmetric and bounded kernel function  $K_{\Sigma}(\mathbf{x})$ . We assume an ML optimization approach with leave-one-out cross-validation strategy.

Suppose that we are given i.i.d. samples  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  drawn from  $p(\mathbf{x})$ . The leave-one-out kernel density estimates are of the form

$$\hat{p}_i(\mathbf{x}) = \frac{1}{N-1} \sum_{j=1, j \neq i}^N K_{\Sigma}(\mathbf{x} - \mathbf{x}_j). \quad (\text{A.35})$$

Recall that these density estimates are easily made consistent and asymptotically unbiased. Using the i.i.d. sample assumption in conjunction with these estimates, the log-likelihood function is determined to be

$$L(\Sigma) = \frac{1}{N} \sum_{i=1}^N \log \hat{p}_i(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N \log \left( \frac{1}{N-1} \sum_{j=1, j \neq i}^N K_{\Sigma}(\mathbf{x}_i - \mathbf{x}_j) \right). \quad (\text{A.36})$$

---

<sup>2</sup> For more results on convergence properties of kernel density estimates, the reader is referred to [12] for an excellent overview of the literature on this subject.

<sup>3</sup> The latter property later becomes appealing when the corresponding kernel density estimator is utilized in conjunction with Renyi's entropy.

From a practical perspective, the optimization of a complete kernel orientation-size matrix  $\Sigma$  might be undesirable in some instances, due to computational complexity [314]. To alleviate this problem, a few simplifications could be assumed.

The data covariance could be diagonalized by a rotation to align the principal components with the axes of the coordinate system. This is achieved by solving for the eigendecomposition of the data covariance matrix as  $\Sigma_{\mathbf{x}} = E[\mathbf{xx}^T] = \mathbf{Q}\Lambda_{\mathbf{x}}\mathbf{Q}^T$ , where  $\mathbf{Q}$  is the orthonormal eigenvector matrix. The data are rotated according to  $\mathbf{z} = \mathbf{Q}^T\mathbf{x}$ . Then a separable (product) kernel can be optimized by individually optimizing the components of the kernel function for each dimension of the data. Specifically, we utilize  $K_{\Lambda}(\mathbf{z}) = K_{\lambda_1}(z_1) \cdots K_{\lambda_n}(z_n)$ . Once the individual kernels are optimized on the marginal components of the rotated data  $\mathbf{z}$ , the kernel for the original data  $\mathbf{x}$  can be obtained simply by rotating the multidimensional kernel obtained through optimization as follows:  $K_{\Sigma}(\mathbf{x}) = K_{\Lambda}(\mathbf{Qz})$ . This procedure effectively aligns the covariance of the kernel function (which is a PDF itself) with the sample covariance of the empirical data of  $\mathbf{x}$ . The spread of the kernel along each eigenvector is also individually optimized using the ML principle by maximizing Eq. (A.36) for each marginal eigendirection in the data.

An even simpler approach is to skip the PCA-based rotation steps of the procedure described above. In this alternative, although the kernels are not aligned to the data covariance, the estimated density estimate will still be unbiased and consistent asymptotically. Further simplification could be achieved by assuming a circular kernel, as is commonly done in the literature, in which case there is only one parameter to optimize [58]. In some cases, the reduction in computational complexity due to this assumption can be preferable to the additional performance gain from a more involved kernel design procedure. For further simplification, one can assume that a circular Gaussian kernel is used and the underlying data distribution is also Gaussian. In this case, according to Silverman's rule-of-thumb, the  $L_2$ -optimal kernel size is given by  $\sigma_* = \text{tr}(\Sigma_{\mathbf{x}})(4/(n+2))^{1/(n+4)}N^{-1/(n+4)} \sim \text{tr}(\Sigma_x)N^{-1/(n+4)}$ , where  $n$  is the data dimensionality [300]. More advanced approximations for  $L_2$ -optimal kernel size selection also exist [170].

From a density estimation point of view, it might be suggested that each kernel has a size that is determined by the local distribution of samples. Specifically, kernel size assignments based on distance to  $k$ th-nearest-neighbor type measures are shown to be efficient in terms of convergence speed [71, 79]. However, from an adaptive signal processing point of view, the computational complexity involved in optimizing one kernel size per sample can be practically prohibitive. Therefore, in most cases the single kernel size scheme is preferred for its simplicity.

### A.3 Nonparametric Entropy Estimation

The problem of entropy estimation appears in many contexts in a variety of fields ranging from basic sciences such as biology [81] and physics [23] to engineering [65]. From a mathematical standpoint, many approaches exist to estimate the differential entropy of a continuous random variable [28]. An obvious approach, usually preferred when there is confidence that the PDF underlying the samples belongs to a known parametric family of PDFs, is to use the samples to estimate the parameters of the specific member of this family, perhaps using maximum likelihood methods, and then to evaluate the entropy of the specific PDF obtained as a result of this procedure. A useful list of explicit Shannon's entropy expressions for many commonly encountered univariate PDFs was compiled by Lazo and Rathie [193]. A similar list for various multivariate PDFs was presented by Ahmed and Gokhale [3]. This approach, although useful in entropy evaluation tasks and effective when the assumed parametric family is accurate, is not competent in adaptation scenarios, where the constantly changing PDF of the data under consideration may not lie in a simple parametric family. Then it becomes necessary to non-parametrically estimate the entropy.

#### Plug-In Estimates for Entropy

The plug-in entropy estimates are obtained by simply inserting a consistent density estimator of the data in place of the actual PDF in the entropy expression. Four types of approaches could be followed when using a plug-in estimate. The first one, named *integral estimates*, evaluates exactly or approximately the infinite integral existing in the entropy definition. Renyi's quadratic entropy estimator (developed and used successfully in this work) belongs to this family of entropy estimators, with an exact evaluation of the integral. An approximate estimate of this type for Shannon's entropy was also proposed [76]. Joe [169] also considered an approximate integral estimate of Shannon's entropy using a kernel-based PDF estimate; however, he concluded that for multivariate cases, the approximate evaluation of the integral becomes complicated. Gyorfi and van der Meulen [125] avoid this problem by substituting a histogram estimate for the PDF.

The second approach, *resubstitution estimates*, further includes the approximation of the expectation operator in the entropy definition with the sample mean. Ahmad and Lin [2] presented a kernel-based estimate for Shannon's entropy of this type and proved the mean-square consistency of this estimate. Joe [169] also considered a similar resubstitution estimate of Shannon's entropy based on kernel PDF estimates, and he concluded that in order to obtain accurate estimates especially in multivariate situations, the number of samples required increased rapidly with the dimensionality of the data. Other examples of this type of entropy estimates are more closely known to the electrical engineering community [34, 62, 325, 346]. These estimates use

spectral estimation–based or polynomial expansion type PDF estimates substituted for the actual PDF in Shannon’s entropy definition, except for the last one, which uses a kernel PDF estimator. In fact, a thorough search of the literature revealed that most estimators known to the electrical engineering community concentrate on resubstitution estimates of Shannon’s entropy. Depending on the specific application the authors are interested in, these estimates are tailored to suit the computational requirements desired from the algorithm. Therefore, it is possible to write out an extensive list of application-oriented references with slight differences in their entropy estimators. The nonparametric estimator for Renyi’s entropy that we derive in Section 2.2 is also a member of the resubstitution class and all theoretical results in these references might apply to it after some minor modifications.

The third approach is called the *splitting data estimate*, and is similar to the resubstitution estimate, except that now the sample set is divided into two parts and one is used for density estimation and the other part is used for the sample mean [126].

Finally, the fourth approach, called *cross-validation estimate*, uses a leave-one-out principle in the resubstitution estimate. The entropy estimate is obtained by averaging the leave-one-out resubstitution estimates of the dataset. Ivanov and Rozhkova [159] proposed such an estimator for Shannon’s entropy using a kernel-based PDF estimator.

## Sample Spacing Estimates for Entropy

In this approach, a density estimate is constructed based on the sample differences. Specifically in the univariate case, if the samples are ordered from the smallest to the largest, one can define the  $m$ -spacing between the samples as the difference between samples that are separated by  $m$  samples in the ordering. This PDF estimate can then be substituted in the entropy definition as in the resubstitution estimates. Surprisingly, although the  $m$ -spacing density estimates might not be consistent, their corresponding  $m$ -spacing entropy estimates might turn out to be (weakly) consistent [23, 27]. The generalization of these estimates to multivariate cases is not trivial, however.

## Nearest Neighbor Estimates for Entropy

For general multivariate densities, the nearest neighbor entropy estimate is defined as the sample average of the logarithms of the normalized nearest neighbor distances plus a constant, named the Euler constant [23]. Kozachenko and Leonenko [187], Tsybakov and van der Meulen [321], and Bickel and Breiman [36] provide different forms of consistency for these estimates under mild conditions on the underlying densities.

## A.4 Estimation of Information-Theoretic Descriptors

The previous section reviewed methods of density of estimation, whereas this section is more specific and deals with the estimation of information-theoretic quantities. The goal is to increase the utility of ITL by providing a better understanding of the information criteria and the parameters associated with their estimators. Here we examine the estimators for Renyi's quadratic entropy and mutual information (QMI<sub>CS</sub> and QMI<sub>ED</sub>) in further detail and present various properties and characteristics that help to define the effects and consequences of the various parameter settings. These results are primarily useful when one wants to estimate the value of these quantities from finite data, but they are also important for cost functions in as much as they provide bounds and help determine the kernel size.

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.

### Renyi's Entropy Estimator

One of the difficulties in comparing or assessing the performance of a system trained with ITL stems from the performance criteria themselves. As we stated, when information-theoretic criteria are computed using Renyi's quadratic entropy with Parzen estimation, the resulting values of entropy have little absolute meaning. They only help to gauge performance in a relative sense when comparing data generated using the same set of parameters, which is still fine for adaptation which is only interested in the extrema of the cost function. Although the actual values change, the location of extrema in parameter space move little, if at all for a large set of kernel sizes. 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  $\sigma$  plays an important role in the convergence of ITL systems and the ability to adapt  $\sigma$  helps to avoid local extremes and assure better convergence [86]. Adjustment of the sample size has also shown 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 Eq. (2.15) has served as the workhorse for much of the recent work in ITL research. Despite its

prevalence, the characteristics of the estimator and its dependencies on the parameters to characterize its performance are mostly experimental. The following sections help to explore the relationships between the entropy estimator and its major parameters by employing a family of curves approach combined with some pertinent examples.

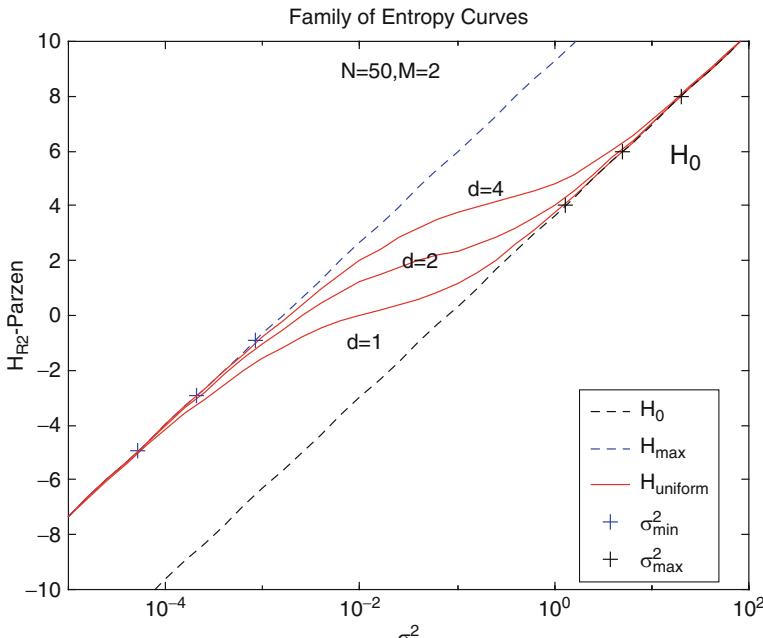
### Families of Entropy Curves

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

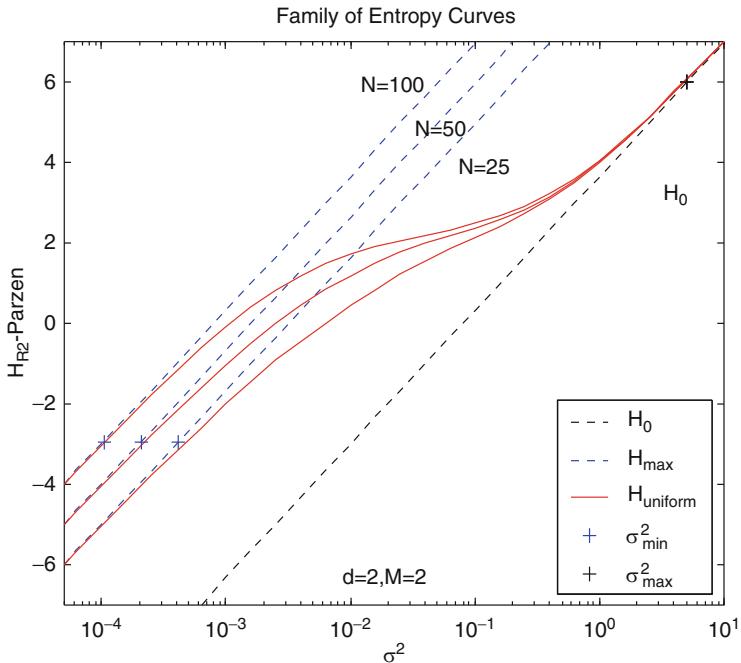
1.  $N$ : The number of data samples or exemplars
2.  $\sigma$ : The kernel size for the Parzen PDF estimate
3.  $M$ : The dimension of the dataset
4.  $d$ : A measure of the extent (or variance) of the data

Figures A.2, A.3, and A.4 illustrate the dependence of the entropy estimate on these parameters and highlight the difficulty with the entropy criterion. 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.

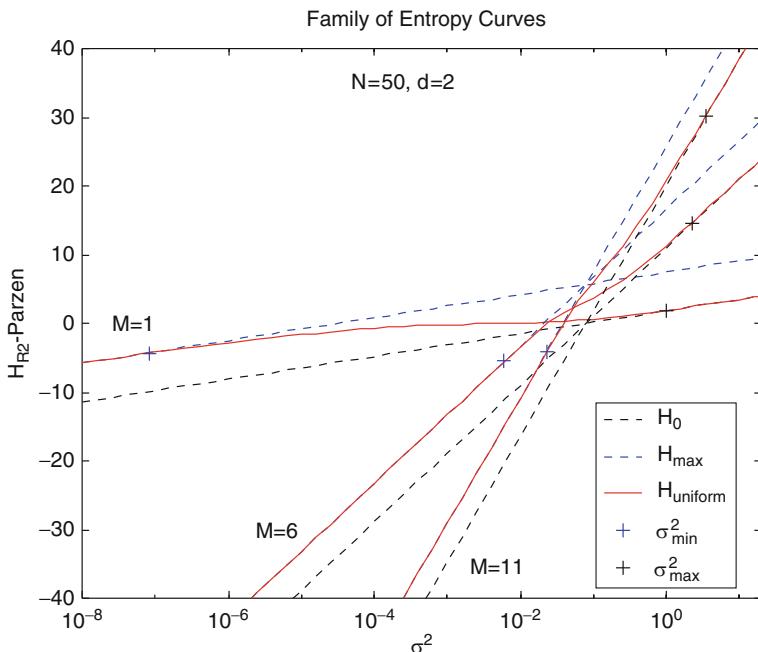
Notice that the entropy value can take on virtually any value depending on the settings of the parameters. Each shows the entropy estimate  $H_2$  from



**Fig. A.2.** Family of entropy curves when the data extent changes (from [223]).



**Fig. A.3.** Family of entropy curves when the number of samples change (from [223]).



**Fig. A.4.** Family of entropy curves when the data dimension changes (from [223]).

Eq. (2.15) in solid lines, versus  $\sigma^2$ , 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.

Also of interest is that in all cases the entropy estimate asymptotically approaches one line labeled  $H_0$  as  $\sigma \rightarrow \infty$  and another a set of parallel line as  $\sigma \rightarrow 0$  for a given set of parameters,  $N$ ,  $M$ , and  $d$ . The first line, denoted  $H_0(\sigma)$ , is a lower bound on  $H_2$  as a function of  $\sigma$  and  $M$ . The second is an upper bound on  $H_2$  as a function of  $\sigma$ ,  $M$ , and  $N$ . The characteristics of these lines can be solved analytically as

$$H_0(\sigma) = \lim_{\sigma \rightarrow \infty} H_2 = \frac{M}{2} \log 4\pi + \frac{M}{2} \log \sigma^2 \quad (\text{A.36})$$

$$H_{\max}(\sigma) = \lim_{\sigma \rightarrow 0} H_2 = \frac{M}{2} \log 4\pi + \frac{M}{2} \log \sigma^2 + \log N, \quad (\text{A.37})$$

Equation (A.37) assumes that the  $N$  data points are all different from each other. In some applications (e.g., discrete or saturated data) some of the data points might be repeated. If this condition exists the maximum equation in Eq. (A.37) can be modified to provide a more accurate maximum as follows;

$$H_{\max}(\sigma) = \lim_{\sigma \rightarrow 0} H_2 = \frac{M}{2} \log 4\pi + \frac{M}{2} \log \sigma^2 + \log \frac{N^2}{N + \sum_{\text{nonunique}} n_i!}, \quad (\text{A.38})$$

where  $n_i$  represents the number of times each data sample is repeated in the dataset. There are some cases where data are acceptable or expected and the adjusted maximum from Eq. (A.38) should be used to determine the maximum entropy for the dataset at hand. However, in other cases, the repeated data might be undesirable, perhaps resulting from the underlying statistics of the dataset and the current state of the mapping function, and the maximum from Eq. (A.37) provides a better basis for comparison with other datasets.

In Eq. (A.36), as  $\sigma \rightarrow \infty$ , 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 Eq. (A.37) as  $\sigma \rightarrow 0$ , a point is reached where there is virtually no interaction between data samples due to 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 solid lines can be interpreted as an approximation of the maximum entropy level that can be achieved at a particular  $\sigma$  for the corresponding  $M$ ,  $N$ , and  $d$ .

It is important to note the distinction between the estimation,  $H_2(\sigma, N)$ , and the actual entropy value,  $H_2(x)$ . The former is a functional estimate of the latter that is dependent on the number of samples  $N$  taken from the random

variable  $\mathbf{X}$  and the kernel size  $\sigma$ , used for PDF estimation. The quantities  $H_0(\sigma)$  and  $H_{\max}(\sigma, N)$  described in Eqs. (A.36–A.38) are also byproducts of the estimation of the actual entropy using Parzen estimation.

Looking at the figures more closely, it is also clear that the properties of Renyi's quadratic entropy with Parzen 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 due to 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 is 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 for Renyi's Entropy

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 endpoints for the kernel annealing process described in [86].

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 Eq. (A.39).

$$\frac{\text{vol}(\text{Sphere}_m)}{\text{vol}(\text{Cube}_m)} = \left( \frac{\sqrt{\pi}}{2} \right) \frac{1}{\Gamma(\frac{m}{2} + 1)}. \quad (\text{A.39})$$

By using Eq. (A.36) to scale the kernel size, the following expression for maximum kernel size is obtained.

$$\sigma_{\max}^2 = \frac{4d^2}{\pi} \Gamma \left( \frac{M}{2} + 1 \right)^{2/M}. \quad (\text{A.40})$$

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 interaction and consequently little benefit to attempt training. Because 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 are highly clustered in the sample space. Although the expression in Eq. (A.40) 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 are concentrated to cover only 10% of the total sample space for relatively tight clustering. Finally, the distance is scaled according to Eq. (A.39) resulting in the expression below for minimum kernel size.

$$\sigma_{\min}^2 = \left( \frac{d}{7N^{1/M}} \right)^2 \frac{4}{\pi} \left[ 0.1 \times \Gamma \left( \frac{M}{2} + 1 \right) \right]^{2/M}. \quad (\text{A.41})$$

The crosses towards the large value of  $\sigma$  in Figures A.2, A.3, and A.4 represent the evaluation of the kernel size maxima and the crosses towards the small values of  $\sigma$  represent minima from Eqs. (A.40) and (A.41), respectively. It is evident from these 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 Eqs. (A.40) and (A.41) 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. Also, if during initial training the dataset 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.

### Scale Invariant Renyi's Entropy

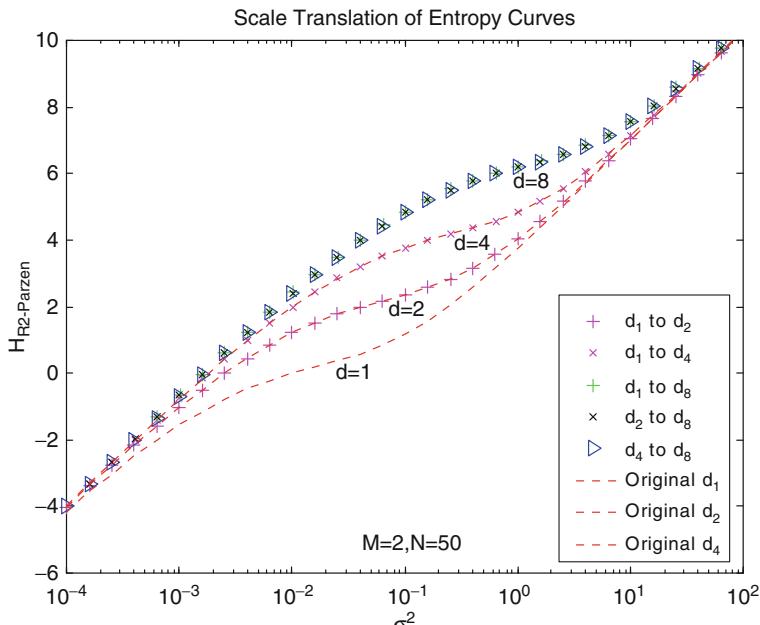
Scale invariance is a desirable characteristic for many cost functions because it allows valid comparisons between datasets without the need for normalization. The ITL entropy criteria, however, do not possess the characteristic of scale invariance. For some problems, preprocessing of the data or the topology of the learning system requires different scaling of the data. In order to fairly compare the entropy of datasets with different scaling a closer inspection of Figure A.2 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 are 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  $\sigma$  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  $(d_1, \sigma_1)$ , to another scale  $(d_2, \sigma_2)$  for comparison, the translation formula in Eq. (A.42) can be applied. The proof follows in the section below.

$$\begin{aligned} H(d_2, \sigma_2) &= H(d_1, \sigma_1) - H_0(\sigma_1) + H_0(\sigma_2) \\ H(d_2, \sigma_2) &= H(d_1, \sigma_1) + M \log \frac{d_2}{d_1} \quad \text{where } \sigma_2 = \frac{d_2}{d_1} \sigma_1. \end{aligned} \quad (\text{A.42})$$

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 because 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 A.5 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 A.2 are translated by various scale factors. The curves denoted “ $d_A$  to  $d_B$ ” use the translation



**Fig. A.5.** Scale translation (Eq. A.42) when the data extent changes (from [223]).

formula in Eq. (A.42) to shift data scaled on the range [0,A] to a range [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.

*Proof of Scaling Equation:* Recall that the Parzen estimation with Gaussian symmetric kernels is,

$$\begin{aligned} H_2(y, \sigma) &= -\log \left[ \frac{1}{N^2} \sum_i \sum_j G(\Delta y_{ij}, 2\sigma^2 I) \right] \\ &= -\log \left[ \frac{1}{N^2} \sum_i \sum_j \frac{1}{(4\pi\sigma^2)^{M/2}} \exp \left( \frac{-(\Delta y_{ij})^T (\Delta y_{ij})}{4\sigma^2} \right) \right]. \quad (\text{A.43}) \end{aligned}$$

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

$$H_2(ay, a\sigma) = -\log \left[ \frac{1}{N^2} \sum_i \sum_j \frac{1}{(4\pi a^2 \sigma^2)^{M/2}} \exp \left( \frac{-(a\Delta y_{ij})^T (a\Delta y_{ij})}{4a^2 \sigma^2} \right) \right]. \quad (\text{A.44})$$

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

$$H_2(ay, a\sigma) = -\log \left[ \frac{1}{|a|^M} \frac{1}{N^2} \sum_i \sum_j \frac{1}{(4\pi\sigma^2)^{M/2}} \exp \left( \frac{-(\Delta y_{ij})^T (\Delta y_{ij})}{4\sigma^2} \right) \right]. \quad (\text{A.45})$$

Utilizing the properties of logarithms of products yields

$$H_2(ay, a\sigma) = -\log \left[ \frac{1}{N^2} \sum_i \sum_j \frac{1}{(4\pi\sigma^2)^{M/2}} \exp \left( \frac{-(\Delta y_{ij})^T (\Delta y_{ij})}{4\sigma^2} \right) \right] - \log \left( \frac{1}{|a|^M} \right). \quad (\text{A.46})$$

and finally, the desired result

$$H_2(ay, a\sigma) = H_{R2}(y, \sigma) + M \log |a|. \quad (\text{A.47})$$

The result in Eq. (A.47) suggests that a scale-invariant criterion based on Renyi's quadratic entropy might be

$$J_2(y, \sigma) = H_2(y, \sigma) - \frac{M}{2} \log [\text{var}(y)] \quad (\text{A.48})$$

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

$$\sigma^2(y) = k \text{ var}(y), \quad (\text{A.49})$$

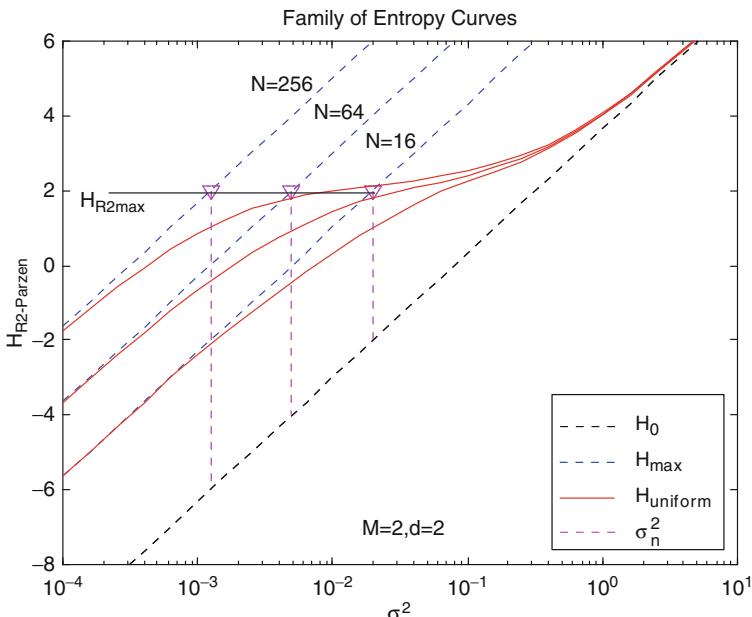
where  $k$  is a constant.

## Sample Size Variations and Batch Size Estimation for Renyi's Entropy

Figure A.3 shows the increase in the entropy estimate with increased sample size for a given  $\sigma$ , implying that use of the entire dataset 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 A.6 shows that this is the case for a family of horizontal lines requiring each to have a unique  $\sigma$  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 Eq. (3.4) with  $\alpha = 2$  as follows,

$$H_{2\max} = -\log \left( \int_{-\infty}^{\infty} u(\mathbf{z})^2 d\mathbf{z} \right) = M \log d. \quad (\text{A.49})$$

It should be noted that analytical solutions for Renyi's entropy only exist in special cases; however, because the uniform distribution provides the



**Fig. A.6.** Family of entropy curves when the training data size changes plotted for the uniform distribution. The right diagonal line is  $H_0$  and the three left ones represent  $H_{\max}$  for different number of samples (from [223]).

maximum entropy for a range-limited signal, it is a useful special case. The resulting kernel sizes correspond to the solution of:

$$H_{2\max} = H_{\max} \quad (\text{A.50})$$

which yields,

$$\sigma_i^2 = \frac{d^2}{4\pi N_i^{2/M}}. \quad (\text{A.51})$$

Equation (A.51) suggests a kernel size as a function of  $N$  that might be used to compare entropy estimates computed with different sample sizes.

Note that Eq. (A.50) equates the maximum value of the Renyi's quadratic entropy from Eqs. (3.4) and (A.49) with the maximum of the estimator in Eqs. (3.12) and (A.37) resulting in a specific kernel size for a given  $N$ ,  $M$ , and  $d$ . Using these kernel sizes, a reasonable metric for comparison of entropy estimates with different samples sizes, that is,  $H_{N_1}(N_1, \sigma_1)$  versus  $H_{N_2}(N_2, \sigma_2)$ , is proposed. Consider the following relative entropy measure:

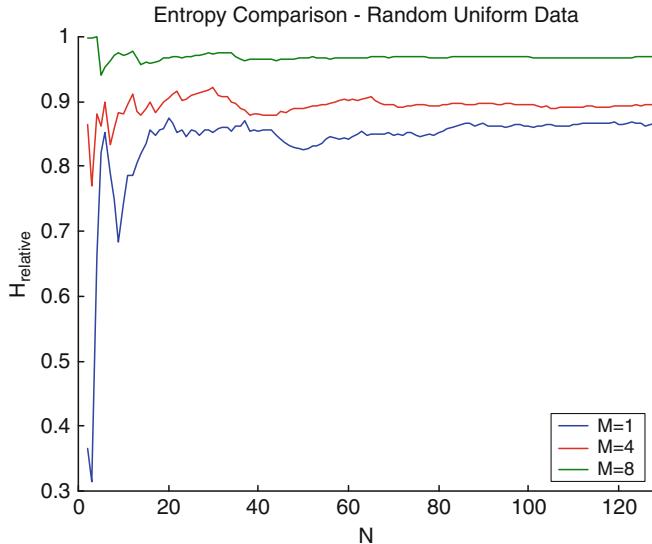
$$H_{\text{relative}}(N_n) = \frac{H_{N_n}(N_n, \sigma_n) - H_0(\sigma_n)}{H_{\max}(\sigma_n) - H_0(\sigma_n)}. \quad (\text{A.52})$$

In Eq. (A.52) the estimated entropy  $H_N$  is computed using the kernel size from Eq. (A.51) 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  $H_{2\max} - H_0$ . 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. Also, if a kernel size other than Eq. (A.51) is used, perhaps due to the spread of the data, as long as the ratio from  $\sigma_1$  to  $\sigma_2$  follows the relationship for  $N$  in Eq. (A.51), valid comparisons can be made.

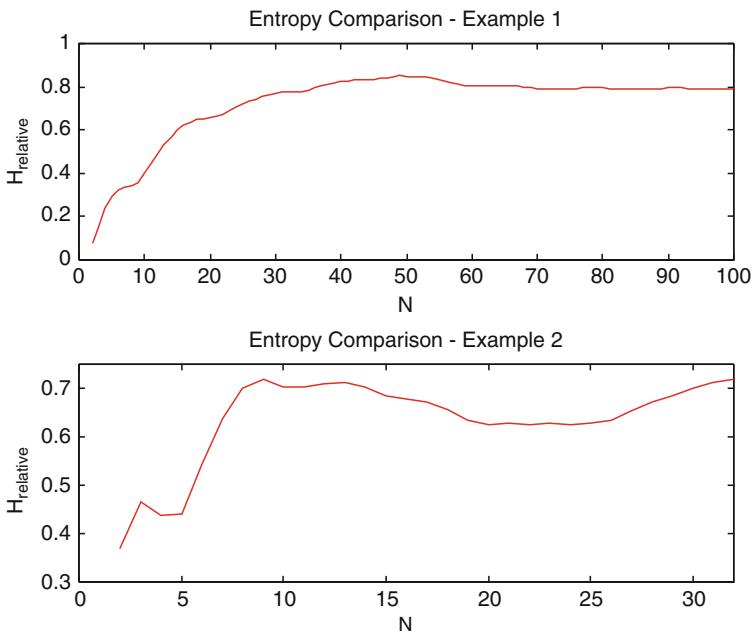
Figure A.7 shows an example of the relative entropy metric for a uniformly distributed random dataset that has been sampled with various values of  $N$ . The curves show the consistency of the relative entropy, aside from the expected fluctuations for very small  $N$ , across a wide range of sample sizes.

Another useful application of the relative entropy metric is motivated by the batch training method described in Chapter 4. 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 below illustrate the approach for several batch-training problems.

Figure A.8 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



**Fig. A.7.** Family of entropy curves plotted with the correction in Eq. A.52 (from [223])



**Fig. A.8.** Relative entropy comparison for two different inputs (sinewave and frequency doubler respectively). Notice that the relative entropy in the most complex example fluctuates much more with the number of samples in the batch (from [223]).

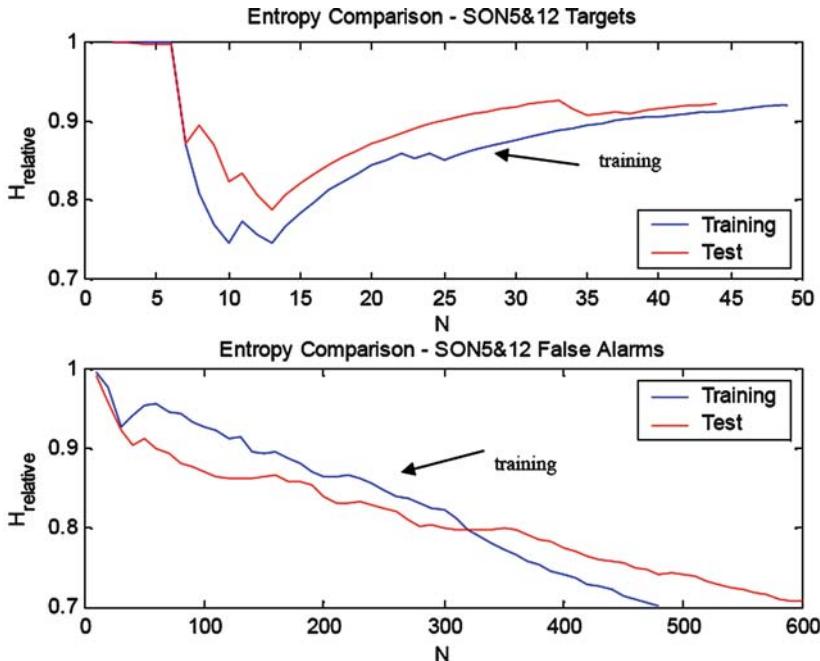


Fig. A.9. Relative entropy comparison—sonar 5 & 12 (from [223]).

with  $N \ll 100$  with little loss of terminal accuracy. The second example is the frequency doubling problem presented in Chapter 5. Recall that in this example, the input dataset consists of  $N = 32$  samples of a composite sinusoidal signal. In this case, the relative entropy reaches a plateau in the  $N = 8$  to  $14$  sample range, suggesting a batch implementation of this size. Previous experimentation with various batch sizes on this problem yielded excellent results for batch sizes of 8 and 16.

Figure A.9 illustrates a third example using a real-world dataset taken from sonar images, the Sonar 5 & 12 feature dataset [223]. This figure shows the relative entropy comparison for the target class and false alarm class for both the training set and the test set. 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.

### Quadratic Mutual Information Estimators

As is the entropy criterion, the QMI<sub>CS</sub> and QMI<sub>ED</sub> estimators for mutual information of Eq. (2.104) are burdened with similar difficulties when comparing

or assessing the performance of systems trained with these criteria. These estimates lack a consistent absolute interpretation, therefore 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 because 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 QMI 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 is 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.

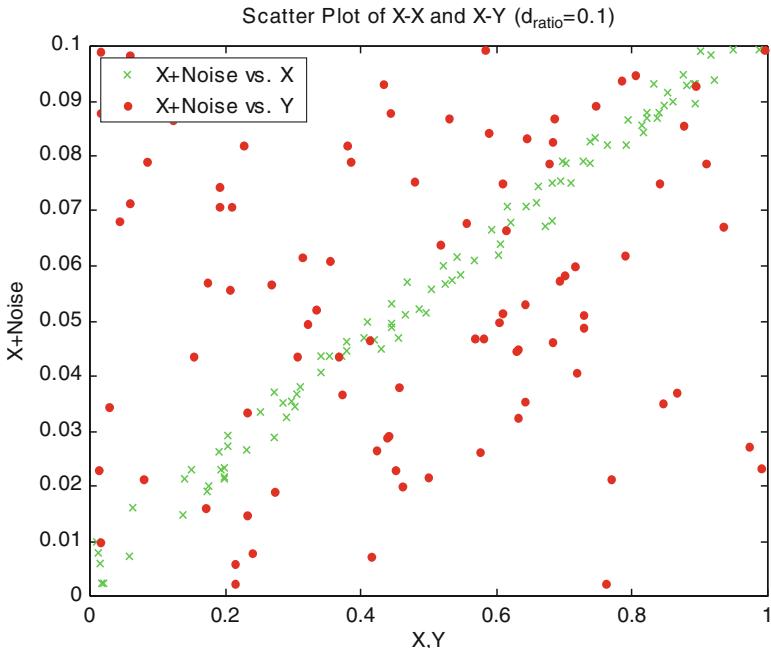
1.  $N$ : The number of data samples or exemplars
2.  $\sigma$ : The kernel size for the Parzen PDF estimate
3.  $M_1, M_2$ : The dimensions of the signals
4.  $d_1, d_2$ : A measure of the extent (or variance) of the signals
5.  $L$ : The number of signals (only investigating  $L = 2$ )

In order to make comparisons with the mutual information estimators, three test datasets are introduced. The first two datasets,  $\mathbf{X}$  and  $\mathbf{Y}$ , are two independent  $N$ -sample datasets based on a random uniform distribution in  $M$ -dimensions. The third dataset,  $\mathbf{Xn}$ , is the  $\mathbf{X}$  dataset with additive Gaussian noise. These datasets are scaled by the ratio,  $d_{\text{ratio}} = d_1/d_2$ , in order to explore the effects of different data scales. These datasets are illustrated graphically in the scatterplot of Figure A.10 for  $M = 1$  and  $d_{\text{ratio}} = 0.1$ . The figure illustrates the expected correlation between  $\mathbf{X}$  and  $\mathbf{Xn}$  and the independence of  $\mathbf{Y}$  and  $\mathbf{Xn}$ .

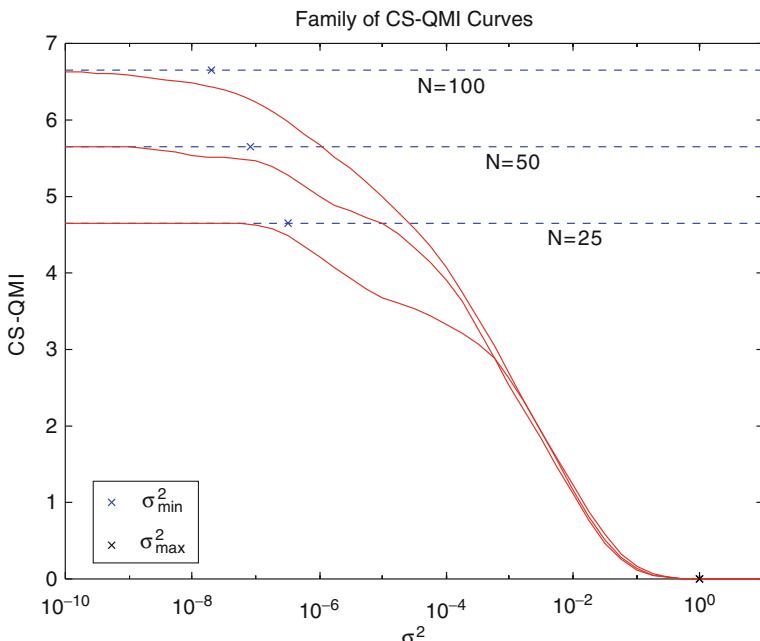
### Effects of Sample Size in QMI Estimators

The first set of curves, in Figures A.11 and A.12, explores the effects of the sample size  $N$  on self-mutual information,  $I(\mathbf{X}, \mathbf{X})$ . Notice first that  $I(\mathbf{X}, \mathbf{X})$  approaches zero for both  $\text{QMI}_{\text{CS}}$  and  $\text{QMI}_{\text{ED}}$  as the kernel size increases to large values. As the kernel size decreases, the saturating limit of  $\text{QMI}_{\text{CS}}$  is defined by Eq. (A.53) as

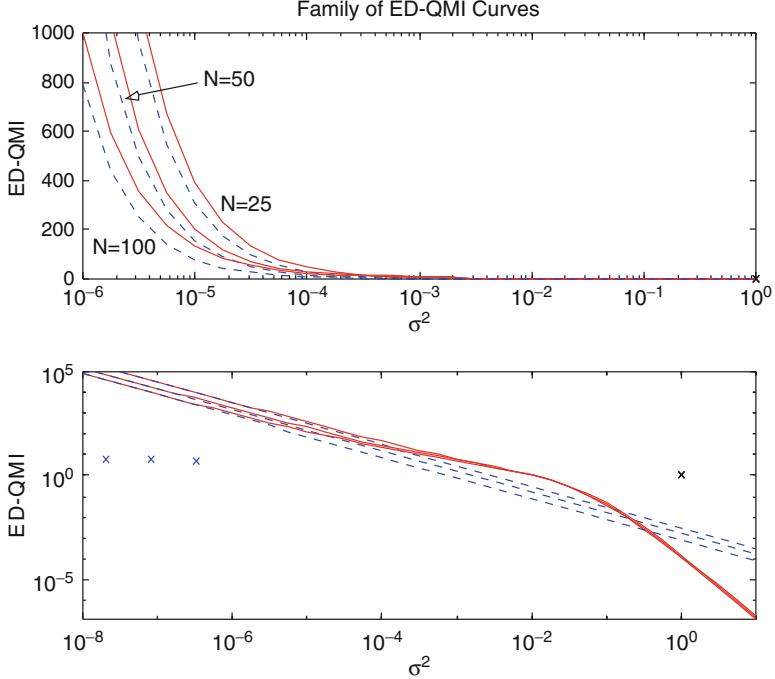
$$I_{\text{CS max}}(L, N) = \lim_{\sigma \rightarrow 0} I_{\text{CS}} = (L - 1) \log N \quad (\text{A.53})$$



**Fig. A.10.** Scatter plot for mutual information test data (from [223]).



**Fig. A.11.** Family of QMI<sub>CS</sub> curves. The QMI for maximum sigma is zero, and its estimated value changes differently as a function of the samples when the kernel size is scanned (from [223]).



**Fig. A.12.** Family of  $QMI_{ED}$  curves for a linear scale (top) and log scale (bottom) (from [223]).

for the case where all  $N$  samples are unique. This saturating limit is illustrated by the horizontal lines in Figure A.11. If some of the data points are repeated, a more complex expression results as described by Eq. (A.54).

$$I_{CS \max}(L, N) = (L - 1) \log \frac{\left( N + \sum_{\text{nonuniquesets}} n_i! \right) \prod_{l=1}^L \left( N + \sum_{\text{nonunique}} n_i! \right)}{\sum_{j=1}^N \prod_{l=1}^L n_{j,l}}. \quad (\text{A.54})$$

In Eq. (A.54) 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.

In the  $QMI_{ED}$  case, as the kernel size decreases, the estimator diverges due to the kernel size term in the denominator of the Gaussian kernels. Figure A.12 shows both linear and logarithmic plots of the  $QMI_{ED}$ . The  $QMI_{ED}$  estimator

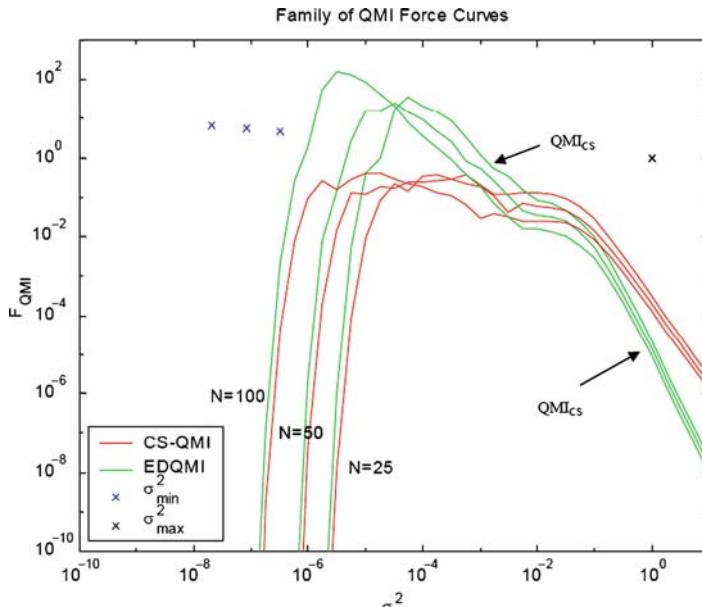
approaches infinity exponentially as described in Eq. (A.55) for  $N$  unique samples and illustrated in Figure A.12 with the straight dashed lines.

$$I_{ED \max}(L, M, N, \sigma) = \lim_{\sigma \rightarrow 0} I_{ED} = \frac{1}{N(4\pi\sigma^2)^{LM/2}} \left( 1 - \frac{1}{N^{L-1}} \right). \quad (\text{A.55})$$

Note that in the  $\text{QMI}_{\text{ED}}$  case, for small kernel sizes, the estimator approaches Eq. (A.55) from above. For nonunique data samples, Eq. (A.56) describes the asymptotes more accurately where the summations and products are the same as in Eq. (A.54).

$$I_{ED \max} = \frac{\frac{1}{N} \left( N + \sum_{\text{nonuniquesets}} n_i! \right) + \frac{1}{N^L} \prod_{l=1}^L \left( N + \sum_{\text{nonunique}} n_i! \right) - 2 \frac{1}{N^L} \sum_{j=1}^N \prod_{l=1}^L n_{j,l}}{N(4\pi\sigma^2)^{LM/2}}. \quad (\text{A.56})$$

In these figures, crosses represent the kernel size extremes described by Eqs. (A.40) and (A.41) for the entropy estimate. Because the components that make up the QMI estimates are essentially based on Renyi's quadratic entropy estimator, these same kernel sizes appear to track the saturation of the QMI estimates as well. The plots in Figure A.13 support this inference further. This figure shows the average information forces as a function of kernel size for the  $\text{QMI}_{\text{CS}}$  and the  $\text{QMI}_{\text{ED}}$ . Notice that for both cases, the information forces roll off in close relation to the kernel size extremes from Eqs. (A.40) and (A.41).



**Fig. A.13.** Family of QMI information forces curves for the CS and ED as a function of the number of samples when the kernel size is scanned (from [223]).

As with the entropy estimators, comparison of mutual information estimates with different sample sizes can be achieved by utilizing the expressions in Eq. (A.53) through (A.56) to develop relative mutual information metrics as

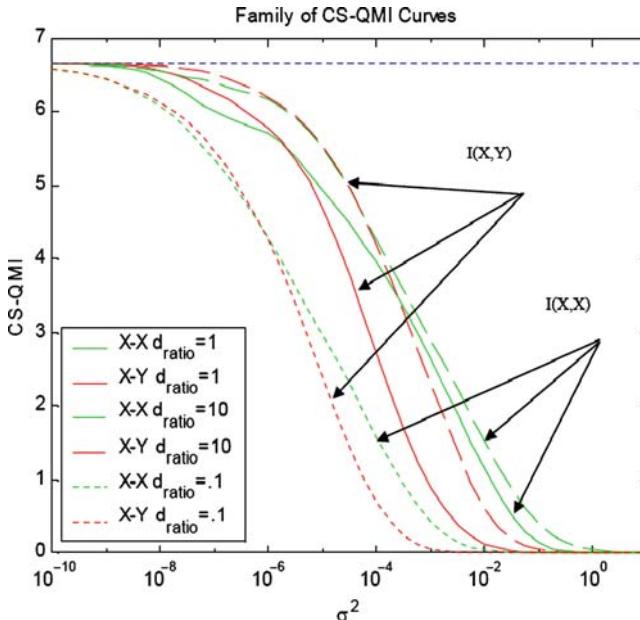
$$I_{CSrelative}(N_n) = \frac{I_{CS}(N_n, \sigma_n)}{I_{CS\max}(N_n)} \quad (\text{A.57})$$

$$I_{EDrelative}(N_n) = \frac{I_{ED\max}(N_n, \sigma_n)}{I_{ED}(N_n, \sigma_n)}, \quad (\text{A.58})$$

where the kernel size  $\sigma_n$  should be adjusted as a function of  $N_n$  per the relationship in (A.51) or (A.38). Note also that Eq. (A.58) is only appropriate for small kernel sizes.

### Effect of Scale in QMI Estimators

The next set of curves examines the dependency of the mutual information estimators on the relative variance of the random variables that are being compared. Figure A.14 shows the mutual information between  $X$  and  $X$ , denoted  $I(X, X)$ , and between  $X$  and  $Y$ , denoted  $I(X, Y)$  for various values of  $d_{ratio}$  where  $N = 100$  and  $M = 1$ . Note here that it is expected that the



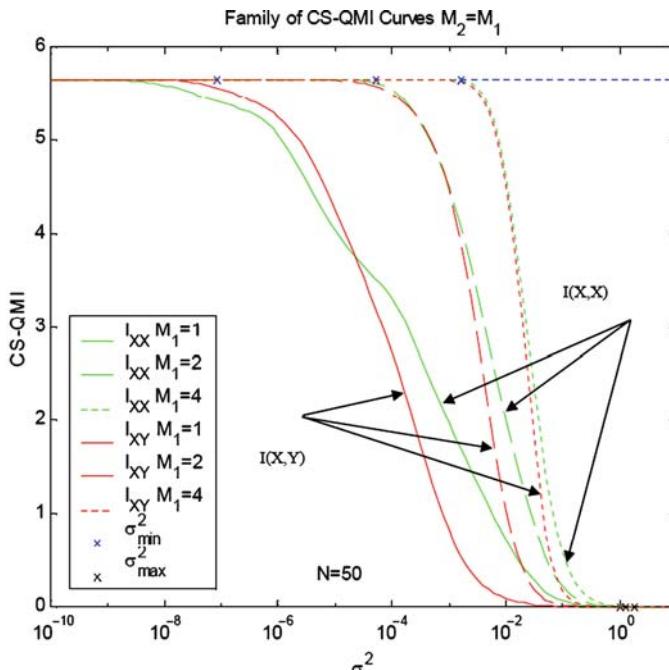
**Fig. A.14.** Family of QMI<sub>CS</sub> curves compared with self mutual information when the dynamic range in the data changes. The estimated QMI( $X, Y$ ) should always be smaller than self QMI, but this only happens for kernels larger than  $10^{-6}$  (from [223]).

self-mutual information  $I(X, X)$  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  $d_{\text{ratio}}$  increases. Although this behavior is undesirable, it is more troubling that for some kernel sizes even when  $d_{\text{ratio}} = 1$ , the mutual information between the uncorrelated variables  $I(X, Y)$  is greater than  $I(X, X)$ . 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 at the proper selection of kernel size. Clearly, when the kernel size is too small, the QMI<sub>CS</sub> can produce unexpected results.

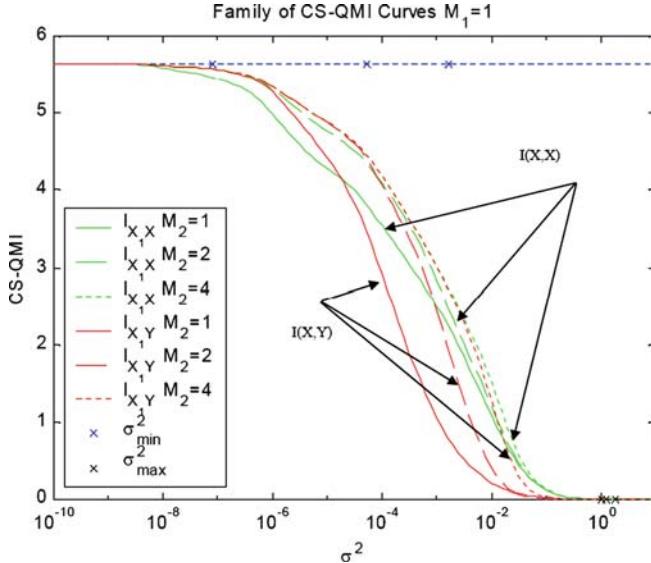
We have experimentally verified that QMI<sub>ED</sub> is better behaved than QMI<sub>CS</sub> in terms of the expectation that  $I(X, X)$  be greater than  $I(X, Y)$ .

### Effect of Dimension in QMI Estimators

Figure A.15 illustrates the effects of increasing dimension on the mutual information estimators. In this figure, the dimensionality of both variables is varied with  $M_1 = M_2$  for values of 1, 2, and 4. For these curves,  $X$  and  $Y$  are independent  $M$ -dimensional random uniform datasets consisting of  $N = 50$  samples. As with the entropy estimator, the slope of the transition increases



**Fig. A.15.** Family of QMI<sub>CS</sub> curves compared with self mutual information when the dimension changes (from [223]).



**Fig. A.16.** Family of QMIs compared with self mutual information when only one of the variables changes dimension (from [223]).

with increasing  $M$ . Note here that the crosses denoting the kernel size limits from Eqs. (A.40) and (A.41) seem to track the saturation points well with increasing  $M$ . Again, note the unexpected performance of the QMIs estimator for small kernel sizes, producing a larger value for  $I(X, Y)$  than for  $I(X, X)$ , as well as the saturation.

In Figure A.16, the effects of increasing dimension on the QMIs mutual information estimator is 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 and  $M_2$  varies from 1 to 4. Here, one variable, either  $X$  or  $Y$ , is an  $M_2$ -dimensional random uniform dataset consisting of  $N = 50$  samples. The other variable,  $X_1$ , is equal to the first dimension of  $\mathbf{X}$  with additive Gaussian noise. So it is expected that the  $I(X_1, X)$  values should be greater than the values for  $I(X_1, Y)$ . 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 Eqs. (A.40) and (A.41) as

$$\sigma_{\max}^2 = \sigma_{\max}^2(\max(M_1, M_2)) \quad (\text{A.59})$$

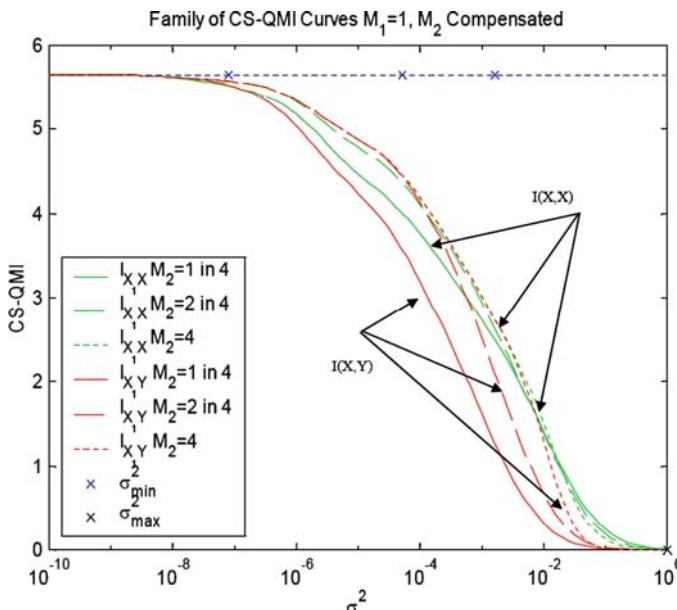
$$\sigma_{\min}^2 = \sigma_{\min}^2(\min(M_1, M_2)). \quad (\text{A.60})$$

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, represented by  $X_1$ , to have the largest value of mutual information. 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 QMI<sub>CS</sub> is again plagued with inverted results for small kernel sizes. For large kernel sizes, all the QMI<sub>CS</sub> curves are greater for  $I(X_1, X)$  than for  $I(X_1, Y)$ . However, the largest value corresponds to the largest dimension rather than the smallest as would be ideal.

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 with  $M_1$  held fixed at one,  $M_2$  fixed at the largest common dimension (which is four), and the intrinsic dimensionality of  $M_2$  varying from 1 to 4.

Figure A.17 shows the results for the QMI<sub>CS</sub>. Notice that the range in which the values for  $I(X_1, X)$  are greater than those for  $I(X_1, Y)$  has increased. Again, for large kernel sizes, all the QMI<sub>CS</sub> curves are greater for



**Fig. A.17.** Family of QMI<sub>CS</sub> curves compared with self mutual information when the estimation is done in the largest dimension space. The range of kernel sizes where the estimation makes sense increases (from [223]).

$I(\mathbf{X}_1, \mathbf{X})$  than for  $I(\mathbf{X}_1, \mathbf{Y})$ . However, in this case, the largest value corresponds to the smallest intrinsic dimension rather as is desired. Although we have only demonstrated the results for QMIs<sub>CS</sub> similar results were obtained for QMIs<sub>ED</sub> perhaps with a slightly larger range of kernel sizes with the expected behavior.

## A.5 Convolution Smoothing

The global convergence theorem for convolution smoothing states that the following optimization problems are equivalent

$$\min_{x \in D \subset \Re^n} g(x) = g(x^*) = \min_{x \in D \subset \Re^n} \hat{g}_\beta(x), \quad \beta \rightarrow 0, \quad (\text{A.61})$$

where the smoothed cost function is defined as

$$\hat{g}_\beta(x) \stackrel{\Delta}{=} g(x) * h_\beta(x) \quad (\text{A.62})$$

and thus both problems result in the global optimal point  $x^*$  [310]. The smoothing functional  $h_\beta(x)$  has to satisfy the following conditions.

$$\text{i.} \quad h_\beta(x) = (1/\beta^n)h(x/\beta), \quad (\text{A.63})$$

$$\text{ii.} \quad \lim_{\beta \rightarrow 0} h_\beta(x) = \delta(x), \quad (\text{A.64})$$

$$\text{iii.} \quad \lim_{\beta \rightarrow 0} \hat{g}_\beta(x) = g(x), \quad (\text{A.65})$$

$$\text{iv.} \quad h_\beta(x) \text{ is a PDF.} \quad (\text{A.66})$$

Condition (iii) guarantees that both  $g(x)$  and  $h_\beta(x)$  are well-behaved functions. Condition (iv) allows the proof techniques from the stochastic optimization literature to be applicable. For our purposes, this strict condition is not necessary, because even if the convolving function does not integrate to one, then the same convolution smoothing effect will be observed, except there will be a scale factor that multiplies the smoothed functional. The most important constraints on the smoothing function are (i) and (ii).

*Conjecture A.1.* Given a specific choice of the kernel function  $\kappa_\sigma(\cdot)$ , there exists a corresponding smoothing functional, which is a solution of

$$\bar{V}_{\alpha,\sigma}(w) = V_\alpha(w) * h_\beta(w), \quad (\text{A.67})$$

where  $V_\alpha(w) = \int p_e^\alpha(e; w) de$ ,  $\bar{V}_{\alpha,\sigma}(w) = \lim_{N \rightarrow \infty} \hat{V}_{\alpha,\sigma}(e)$  and that satisfies the conditions (i)–(iv) given above.

*Support:* There are a number of theoretical and experimental observations that support this conjecture. First, consider the nonparametric information potential we use for the error samples.

$$\hat{V}_{\alpha,\sigma}(e) = \frac{1}{N^\alpha} \sum_j \left( \sum_i \frac{1}{\sigma} \kappa_\sigma \left( \frac{e_j - e_i}{\sigma} \right) \right)^{\alpha-1} = \frac{1}{\sigma^{\alpha-1}} \hat{V}_{\alpha,1}(e/\sigma). \quad (\text{A.68})$$

Notice that an increase in kernel size causes a dilation in the  $e$ -space. Therefore all points, including all local extremes, move radially away from the origin when  $\sigma$  is increased. The only point that maintains its position is the origin. From this, we conclude that if the function approximator (adaptive system) belongs to the function class being approximated (i.e., the error at the optimal point is zero), then the location of the global solution is independent of the kernel size. Moreover, if the function approximator used is a contractive mapping, which is the case in feedforward neural networks, for example, then the dilation in the  $e$ -space must be followed by dilation in the weight-space, hence the volume of the domain of attraction of the global optimum is increased.

In addition, consider the asymptotic behavior of the nonparametric information potential estimator. Because Parzen windowing is a consistent estimator, as the number of samples goes to infinity, the estimated PDF converges to the actual PDF convolved with the selected kernel function [241], which also is valid in the mean; that is,

$$\bar{V}_{\alpha,\sigma}(w) = \int [p_e(e; w)_e^* \kappa_\sigma(e)]^\alpha de, \quad (\text{A.69})$$

where  $*$  denotes a convolution with respect to the variable  $e$ . Equating (A.69) to the convolution of the true information potential  $V_\alpha(w)$  and the (hypothetical) smoothing functional  $h_\beta(w)$ , we obtain the condition Eq. (A.67). Consider the explicit form of this equality written in terms of the kernel function and the error PDF.

$$h_\beta(w)_w^* \int p_e^\alpha(e; w) de = \int [p_e(e; w)_e^* \kappa_\sigma(e)]^\alpha de. \quad (\text{A.70})$$

Taking the Laplace transform of both sides with respect to  $w$ , we can isolate the Laplace transform of  $h_\beta(w)$  in terms of the transforms of the remaining quantities. The Laplace transform of  $h_\beta(w)$  is guaranteed to exist if the error PDF and the kernel function are absolutely integrable functions and  $\alpha \geq 1$ , which is the case. We can write this function in the transform domain as the following ratio.

$$H_\beta(s) = \frac{L_w \int [p_e(e; w)_e^* \kappa_\sigma(e)]^\alpha de}{L_w \int p_e^\alpha(e; w) de} = \frac{\int L_w [p_e(e; w)_e^* \kappa_\sigma(e)]^\alpha de}{\int L_w [p_e^\alpha(e; w)] de}. \quad (\text{A.71})$$

The right-hand side is a function of  $\sigma$  only, because the integration over  $e$  from  $-\infty$  to  $\infty$  eliminates this variable.

Because  $H_\beta(s)$  exists,  $h_\beta(w)$  must be absolutely integrable, therefore,  $\lim_{w \rightarrow \pm\infty} h_\beta(w) = 0$ . We next observe that as  $\sigma \rightarrow 0$ , the numerator of Eq. (A.71) converges to the denominator, hence the bandwidth of  $H_\beta(\omega)$  (considering the Fourier transform) increases. An increase in frequency-domain bandwidth is accompanied by a decrease in duration of the impulse response in the time domain, thus the width of  $h_\beta(w)$  decreases as  $\sigma \rightarrow 0$ ; note that there is a nonlinear monotone relation between  $\beta$  and  $\sigma$ .

Now that we know the width of  $h_\beta(w)$  decreases monotonically as  $\sigma \rightarrow 0$ , that it is always absolutely integrable, and that it converges to  $\delta(w)$  in the limit, we conclude that it has to be unimodal, symmetric, and positive for all  $w$ . Consequently, even if  $h_\beta(w)$  does not integrate to 1, it integrates to some finite value and therefore it is a scaled version of a PDF. A scaling factor in the convolution process does not affect the nature of the smoothing but only the scale factor of the smoothed performance surface.

Although it is not easy to solve for the corresponding smoothing function from Eq. (A.67), we showed that the solution still satisfies some of the required conditions, specifically (ii)–(iv). Furthermore, the dilation in the  $e$ -space, presented in Eq. (A.68), hints towards the validity of condition (i). However, it has not been possible to verify that the first condition is satisfied in general for any mapper, nor it was possible to set forth the conditions under which this occurs. Therefore, we propose the existence of a smoothing functional corresponding to each kernel choice as a conjecture.

The second issue in the application of kernel annealing to real problems is to define the rate of annealing to guarantee that the optimal solution is obtained. The problem is formally very similar to the temperature annealing schedule in simulated annealing [275], except that here one cannot easily evaluate the probability of being caught in a local minimum. Therefore, the annealing rate is left to the experimenter.

Consequently, we propose the following methodology to achieve global optimization when using the current nonparametric entropy estimator in EEC training of adaptive systems. Start with a large kernel size, and during the adaptation gradually and slowly decrease it towards a predetermined suitable value established by the Simpson rule or equivalent; the local solutions, which would trap the training for those same initial conditions when the  $w$ -space has not been dilated, will be avoided. Hence, global optimization will be achieved still using a gradient descent approach. In Chapter 6 we experimentally show that good solutions are found reliably.

As a final remark, we also showed that as the kernel size in the estimator is increased, the entropy estimate approaches the log of a scaled version of the sample variance, thus in EEC error entropy and error variance become asymptotically identical.

## A.6 Conclusions

This appendix reviews basic concepts of density estimation, the ITL estimators for Renyi's quadratic entropy and quadratic mutual information, and a brief overview of convolution smoothing. The purpose of the density estimation review is to provide pointers to the literature and review some basic concepts that may be helpful to estimate the kernel size which is so important in ITL.

The material on the ITL estimators of IP and QMI illustrates using the family of curves approach to how the estimators of these quantities change as a function of the values selected in practical conditions for kernel size and number of samples. We can clearly see the wide variability of the estimated values but of importance is the saturation effect achieved for large and small kernel sizes. Finally we present the theory of convolution smoothing and our present understanding of its applicability for the kernel size annealing during adaptation. Unfortunately the annealing rate has not been determined; there is no formal proof of the link between convolution smoothing and kernel size annealing. This may be achieved in future research by our group or by some of the readers.

---

## Bibliography

1. Aczél J., Daróczy Z., On measures of information and their characterizations, *Mathematics in Science and Engineering*, vol. 115, Academic Press, New York, 1975.
2. Ahmad I., Lin P., A nonparametric estimation of the entropy for absolutely continuous distributions, *IEEE Trans. on Inf. Theor.*, 22:372–375, 1976.
3. Ahmed N., Gokhale D., Entropy expressions and their estimators for multivariate distributions, *IEEE Trans. on Inf. Theor.*, 35:688–692, 1989.
4. Al-Naffouri T., Zerguine A., Bettayeb M., A unifying view of error nonlinearities in LMS adaptation, in *Proc. ICASSP*, vol. III, Seattle, pp. 1697–1700, May 1998.
5. Amari S., Cichocki A., Yang H., A new learning algorithm for blind signal separation. *Advances in Neural Information Processing Systems*, vol. 8 pp. 757–763, MIT Press, Cambridge, MA, 1996.
6. Amari S., Nagaoka H., Methods of information geometry, *Mathematical Monographs*, vol. 191, American Mathematical Society, Providence RI, 2000.
7. Aronszajn N., The theory of reproducing kernels and their applications, *Cambridge Philos. Soc. Proc.*, vol. 39:133–153, 1943.
8. Aronszajn N., Theory of reproducing kernels, *Trans. of the Amer. Math. Soc.*, 68(3):337–404, 1950.
9. Atick J., Redlich A., Towards a theory of early visual processing, *Neural Comput.*, 2:308–320, 1990.
10. Attneave F., Some informational aspects of visual perception, *Psychol. Rev.*, 61; 183–193, 1954.
11. Babich G., Camps O., Weighted Parzen windows for pattern classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(5):567–570, 1996.
12. Bach F., Jordan M., Kernel independent component analysis, *J. Mach. Learn. Res.*, 3:1–48, 2002.
13. Bach F., Jordan M., Finding clusters in independent component analysis, in *Int. Symposium on Independent Component Analysis and Blind Signal Separation*, Nara, Japan, pp. 891–896, 2003.
14. Baldi P., Hornik K., Neural networks and principal component analysis: learning from examples without local minima, *Neural Netw.*, 1:53–58, 1989.
15. Bagshaw P., Paul Bagshaw’s database for evaluating pitch determination algorithms. Available online at <http://www.cstr.ed.ac.uk/research/projects/fda>.

16. Barlow H., Unsupervised learning. *Neural Comput.*, 1(3):295–311, 1989.
17. Barlow H., Kausal T., Mitchison G., Finding minimum entropy codes, *Neural Comput.*, 1(3):412–423, 1989.
18. Basilevsky A., Statistical Factor Analysis and Related Methods: Theory and Applications, John Wiley, New York, 1994.
19. Basu A., Lindsay B. Minimum disparity estimation in the continuous case: Efficiency, distributions, robustness, *Ann. Inst. Statist. Math.*, 46:683–705, 1994.
20. Battiti R., First and second order methods for learning: Between steepest descent and Newton's method, *Neural Comput.*, 4(2):141–166, 1992.
21. Battiti R., Using mutual information for selecting features in supervised neural net learning, *IEEE Trans. Neural Netw.*, 5(4):537–550, July 1994.
22. Beadle E., Schroeder J., Moran B., An overview of Renyi's entropy and some potential applications, *42<sup>nd</sup> Asilomar Conference on Signals, Systems and Computers*, October 2008.
23. Beck C., Schlogl F., *Thermodynamics of Chaotic Systems*, Cambridge University Press, Cambridge, UK, 1993.
24. Becker S., Hinton G., A self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355:161–163, 1992.
25. Becker. S., Unsupervised learning with global objective functions. In M. A. Arbib, (Ed.), *The Handbook of Brain Theory and Neural Networks*, pp. 997–1000. MIT Press, Cambridge, MA, 1998c
26. Bell A., Sejnowski T., An information-maximization approach to blind separation and blind deconvolution. *Neural Comput.*, 7(6):1129–1159, 1995.
27. Beirlant J., Zuijlen M., The empirical distribution function and strong laws for functions of order statistics of uniform spacings, *J. Multiva. Anal.*, 16:300–317, 1985.
28. Beirlant J., Dudewicz E., Gyorfi L., van der Meulen E., Nonparametric entropy estimation: An overview, *Int. J. Math. Statist. Sci.*, 6(1):17–39, 1997.
29. Belouchrani A., Abed-Meraim K., Cardoso J., Moulines E., A blind source separation technique using second-order statistics, *IEEE Trans. Signal Process.*, 45(2):434–444, 1997.
30. Ben-Hur A., Horn D., Siegelmann H., Vapnik V., Support vector clustering, *J. Mach. Learn. Res.*, 2:125–137, 2001.
31. Benedetto S., Biglieri E., Nonlinear equalization of digital satellite channels, *IEEE J. Select. Areas Commun.*, 1:57–62, Jan 1983.
32. Bengtsson I., Zyczkowski K., *Geometry of quantum states*, Cambridge, UK, 2006.
33. Benveniste A., Goursat M., Ruget G., Robust identification of a non-minimum phase system: Blind adjustment of a linear equalizer in data communications, *IEEE Trans. Autom. Control*, 25(3):385–399, 1980.
34. Bercher J., Vignat C., Estimating the Entropy of a Signal with Applications, *IEEE Trans. Signal Process.*, 48(6):1687–1694, 2000.
35. Berlinet A., Thomas-Agnan C., *Reproducing Kernel Hilbert Spaces in Probability and Statistics*, Kluwer, Norwell, MA, 2003.
36. Bickel P., Breiman L., Sums of functions of nearest neighbor distances, moment bounds, limit theorems and a goodness-of-fit test, *Ann. Statist.*, 11:185–214, 1983
37. Biem A., Katagiri S., Juang B., Pattern recognition using discriminative feature extraction, *IEEE Trans. Signal Process.*, 45(2):500–504, Feb. 1997.

38. Bishop C., *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford, 1995.
39. Bhattacharyya A., On a measure of divergence between two statistical populations defined by their probability distributions, *Bul. Calcutta Math. Soc.*, 35:99–109, 1943.
40. Bourbaki N., *Topological Vector Spaces*, Springer, 1987
41. Bregman L.M. (1967). The relaxation method of finding the common points of convex sets and its application to the solution of problems in convex programming. *USSR Comput. Math. Phys.* 7:200–217.
42. Brown J., Puckette M., Calculation of a “narrowed” autocorrelation function, *J. Acoust. Soc. Am.*, 85:1595–1601, 1989.
43. Burbea J., Rao C, Entropy differential metric, distance and divergence measures in probability spaces: A unified approach, *J. Multivar. Anal.*, 12:575–596, 1982.
44. Campbell L., A coding theorem and Renyi’s entropy, *Inf. Control*, 8:423–429, 1965
45. Candocia F., Principe J., Super-resolution of images based on local correlations, *IEEE Trans. Neural Netw.*, 10(2):372–380, 1992.
46. Cardoso J., Souloumiac A., Blind beamforming for non-Gaussian signals, *Radar Signal Process., IEE Proc. F*, 140(6):362–370, December 1993.
47. Cardoso J., Blind signal separation: Statistical principles, *Proc. IEEE*, 86(10):2009–2025, 1998.
48. Carnell, A., Richardson, D., Linear algebra for time series of spikes. In: *Proc. European Symp. on Artificial Neural Networks*, pp. 363–368. Bruges, Belgium (2005)
49. Carreira-Perpinan M., Mode-finding for mixtures of Gaussian distributions, *IEEE Trans. Pattern Anal. Mach. Inte.*, 22(11):1318–1323, November 2000.
50. Carreira-Perpinan M., Gaussian mean shift is an EM algorithm, *IEEE Trans. Pattern Anal. Mach. Inte.*, 29(5):767–776, 2007.
51. Casals J., Jutten C., Taleb A., Source separation techniques applied to linear prediction, *Proc. ICA’00*, Helsinki, Finland, pp. 193–204, 2000.
52. Chechik G., Tishby N., Temporal dependent plasticity: An information theoretic account, *Proc. Neural Inf. Process. Syst.*, 13:110–116, 2001.
53. Chen B., Hu J., Pu L., Sun Z., Stochastic gradient algorithm under  $(h, \phi)$ -entropy criterion, *Circuits Syst. Signal Process.*, 26:941–960, 2007.
54. Chen Z., Haykin S., Eggermont J., Bekker S., Correlative learning: A basis for brain and adaptive systems, John Wiley, Hoboken, NJ, 2007.
55. Cheng Y., Mean shift, mode seeking and clustering, *IEEE Trans. Pattern Anal. Mach. Inte.*, 17(8):790–799, August 1995.
56. Chernoff H., A measure of asymptotic efficiency of tests for a hypothesis based on a sum of observations. *Ann. Math. Stat.*, 23:493–507, 1952.
57. Chi C., Chen C., Cumulant-based inverse filter criteria for MIMO blind deconvolution: Properties, algorithms, and application to D/CDMA systems in multipath, *IEEE Trans. Signal Process.*, 49(7):1282–1299, 2001
58. Chiu S., Bandwidth selection for kernel density estimation, *Ann Statist.*, 19:883–905, 1991.
59. Choi S., Cichocki A., Amari S., Flexible independent component analysis, *J. VLSI Signal Process.*, 26:25–38, 2000.

60. Comaniciu D., Ramesh V., Meer P., Real-time tracking of nonrigid objects using mean shift, in *Proceedings of IEEE Conf. Comput. Vision and Pattern Recogn.*, 2:142–149, June 2000.
61. Comaniciu D., Meer P., Mean shift: A robust approach toward feature space analysis, *IEEE Trans. Pattern Anal. Mach. Inte.*, 24(5):603–619, May 2002.
62. Comon P., Independent component analysis, a new concept?, *Signal Process.*, 36(3):287–314, 1994.
63. Cortez C., Vapnik V., Support vector networks. *Mach. Learn.*, 20:273–297, 1995.
64. Cover T., Classification and generalization capabilities of linear threshold units, Rome air force technical documentary report RADC-TDR-64-32, Tech. Rep., Feb 1964.
65. Cover T., Thomas J., *Elements of Information Theory*, Wiley, New York, 1991
66. Csiszar I., Information type measures of difference of probability distributions and indirect observations, *Studia Sci. Math. Hungaria*, 2: 299–318, 1967.
67. Dayan, P. Abbott L.F. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. MIT Press, Cambridge, MA, 2001.
68. Deco G., Obradovic D., *An Information-Theoretic Approach to Neural Computing*, Springer, New York, 1996.
69. DeFigueiredo R., A generalized Fock space framework for nonlinear system and signal analysis, *IEEE Trans. Circuits Syst.*, CAS-30(9):637–647, Sept. 1983.
70. Dempster, A., A generalization of Bayesian inference, *J. Roy. Statist. Soc.*, B, 30:205–247, 1968.
71. Devroye L., *A Course on Density Estimation*, Birkhauser, Boston, 1987.
72. Dhillon I., Guan Y., Kulisweifeng B., Kernel k-means, spectral clustering and normalized cuts”, *Proc. Tenth ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining (KDD)*, pp. 551–556, August 2004.
73. Di Marzio M., Taylor C., Kernel density classification and boosting: An L2 analysis. *Statist Comput.*, 15(2):113–123, 2004.
74. Diggle P., Marron J.S., Equivalence of smoothing parameter selectors in density and intensity estimation. *J. Am. Statist. Assoc.* 83(403):793–800, 1988.
75. Ding H., He X., Zha H., Gu M., Simon H., A min-max cut algorithm for graph partitioning and data clustering. In *Proc. IEEE Int. Conf. Data Mining*, pp. 107–114, San Jose, CA, November 29–December 2, 2001.
76. Dmitrev Y., Tarasenko F., On the estimation functions of the probability density and its derivatives, *Theor. Probab. Appl.*, 18:628–633, 1973.
77. Donoho D., On minimum entropy deconvolution, in *Applied Time Series Analysis II*, Academic Press, New York, 1981, pp. 565–609.
78. Douglas S., Meng H., Stochastic gradient adaptation under general error criteria, *IEEE Trans. Signal Process.*, 42:1335–1351, 1994.
79. Duda R., Hart P., *Pattern Recognition and Scene Analysis*, Wiley, New York, 1973.
80. Duda R., Hart P., Stork D., *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, 2nd edition, 2001.
81. Durbin R., Eddy S., Krogh A., Mitchinson G., *Biological Sequence Analysis*, Cambridge University Press, Cambridge, UK, 1998.
82. Duttweiler D., Kailath T., RKHS approach to detection and estimation problems—part IV: Non-Gaussian detection, *IEEE Trans. Inf. Theor.*, IT-19(1):19–28, Jan. 1973.

83. Duttweiler D., Kailath T., RKHS approach to detection and estimation problems—part V: Parameter estimation, *IEEE Trans. Inf. Theor.*, IT-19(1):29–37, Jan. 1973.
84. Edmonson W., Srinivasan K., Wang C., Principe J. A global least square algorithm for adaptive IIR filtering, *IEEE Trans. Circuits Syst.*, 45(3):379–384, 1996.
85. Epanichnikov V., Nonparametric estimation of a multivariate probability density, *Theory Prob. Appl.* 14:153–158, 1969.
86. Erdogmus D., Information theoretic learning: Renyi's entropy and its applications to adaptive systems training, Ph.D. Dissertation, University of Florida, Gainesville, 2002.
87. Erdogmus D., Principe J.C., An error-entropy minimization algorithm for supervised training of nonlinear adaptive systems, *IEEE Trans. Signal Process.*, 50(7):1780–1786, 2002.
88. Erdogmus D., J. Principe, Generalized information potential for adaptive systems training, *IEEE Trans. Neural Netw.*, 13(5):1035–1044, 2002.
89. Erdogmus D., Principe J., Hild II K., Do Hebbian synapses estimate entropy?, *Proc. IEEE Workshop on Neural Networks for Signal Process.*, Martigny, Switzerland, pp. 199–208, 2002.
90. Erdogmus D., Principe J., Kim S., Sanchez J., A recursive Renyi's entropy estimator, *Proc. IEEE Workshop on Neural Networks for Signal Process.*, Martigny, Switzerland, pp. 209–217, 2002.
91. Erdogmus D., Hild K., Principe J., Beyond second order statistics for learning: a pairwise interaction model for entropy estimation, *J. Natural Comput.*, 1(1):85–108, 2003.
92. Erdogmus D., Principe J., Lower and upper bounds for misclassification probability based on Renyi's information, *J. VLSI Signal Process. Syst.*, 37(2/3):305–317, 2004.
93. Erdogmus D., Hild II K., Lazaro M., Santamaria I., Principe J., Adaptive blind deconvolution of linear channels using Renyi's nntropy with Parzen estimation, *IEEE Trans. Signal Process.*, 52(6):1489–1498, 2004.
94. Erdogmus D., Agrawal R., Principe J., A mutual information extension to the matched filter, *Signal Process.*, 85(5):927–935, May 2005.
95. Erdogmus D., Principe J. From linear adaptive filtering to nonlinear signal processing” *IEEE SP Mag.*, 23:14–33, 2006.
96. Erdogmus D., Ozertem U., Self-consistent locally defined principal surfaces. In *Proc. Int. Conf. Acoustic, Speech and Signal Processing*, volume 2, pp. 15–20, April 2007.
97. Fano R., *Transmission of Information: A Statistical Theory of Communications*, MIT Press, New York, 1961.
98. Feder M., Merhav N., Relations between entropy and error probability, *IEEE Trans. Inf. Theor.*, 40(1): 259–266, 1994.
99. Feng X., Loparo K., Fang Y., Optimal state estimation for stochastic systems: An information theoretic approach, *IEEE Trans. Autom. Control*, 42(6):771–785, 1997.
100. Fine S., Scheinberg K., Cristianini N., Shawe-Taylor J., Williamson B., Efficient SVM training using low-rank kernel representations, *J. Mach. Learn. Res.*, 2:243–264, 2001.
101. Fisher R., The use of multiple measurements in taxonomic problems, *Ann. Eugenics* 7; 170–188, Wiley, New York, 1950.

102. Fisher J., Ihler A., Viola P., Learning informative statistics: A nonparametric approach, *Proceedings of NIPS'00*, pp. 900–906, 2000.
103. Fock V., *The Theory of Space Time and Gravitation*”, Pergamon Press, New York, 1959.
104. Fox J., *An R and S Companion to Applied Regression*, Sage, London, 2002.
105. Friedman J., Tukey J., A Projection Pursuit Algorithm for Exploratory Data Analysis, *IEEE Trans. Comput., Ser. C*, 23:881–889, 1974.
106. Fu K., Statistical pattern recognition, in *Adaptive, Learning and Pattern Recognition Systems*, Mendel and Fu Eds., Academic Press, New York, 1970, pp. 35–76.
107. Fukumizu K., Gretton A., Sun X., Scholkopf B., Kernel measures of conditional dependence. In Platt, Koller, Singer, and Roweis Eds., *Advances in Neural Information Processing Systems 20*, pp. 489–496. MIT Press, Cambridge, MA, 2008.
108. Fukunaga K., *An Introduction to Statistical Pattern Recognition*, Academic Press, New York, 1972
109. Fukunaga K., Hostetler L., The estimation of the gradient of a density function with applications in pattern recognition, *IEEE Trans. Inf. Theor.*, 21(1):32–40, January 1975.
110. Gersho A., Gray R. *Vector Quantization and Signal Compression*. Springer, New York, 1991
111. Gdalyahu Y., Weinshall D., Werman M., Self-organization in vision: Stochastic clustering for image segmentation, perceptual grouping, and image database organization. *IEEE Trans. Pattern Anal. Mach. Inte.*, 23(10):1053–1074, 2001.
112. Girolami M., Orthogonal series density estimation and the kernel eigenvalue problem. *Neural Comput.*, 14(3):669–688, 2002.
113. Glass L., Mackey M., From clock to chaos, Princeton University Press, Princeton, NJ, 1998.
114. Gokcay E., Principe J., Information theoretic clustering, *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(2):158–171, 2002.
115. Grossberg S., Competitive learning: From interactive activation to adaptive resonance, in *Connectionist Models and Their Implications: Readings from Cognitive Science* (Waltz, D. and Feldman, J. A., Eds.), Ablex, Norwood, NJ, pp. 243–283, 1988.
116. Golub G., Van Loan C., *Matrix Computation*, 3rd ed. The Johns Hopkins University Press, Baltimore, Maryland, 1996.
117. Gonzalez T., Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.
118. Granas A., Dugundji J., *Fixed Point Theory*, Springer-Verlag, New York, 2003.
119. Granger C., Maasoumi E., and Racine J., A dependence metric for possibly nonlinear processes. *J. Time Series Anal.*, 25(5):649–669, 2004.
120. Grassberger, P., I. Procaccia, Characterization of strange attractors, *Phys. Rev. Lett.*, 50(5):346–349, 1983.
121. Greengard L., Rokhlin V., A fast algorithm for particle simulations. *J. Comput. Phys.*, 73(2):325–348, 1987.
122. Greengard L., Strain J., The fast Gauss transform. *SIAM J. Sci. Statist. Comput.*, 12(1):79–94, 1991.
123. Gretton, A., Herbrich R., Smola A., Bousquet O., Schölkopf B., Kernel Methods for Measuring Independence,” *J. Mach. Learn. Res.*, 6:2075–2129, 2005.

124. Gretton A., Borgwardt K., Rasch M., Scholkopf B, Smola A., A kernel method for the two-sample-problem, in *Advances in Neur. Inf. Process. Syst. (NIPS)*. MIT Press, Cambridge, MA, 2006.
125. Gyorfi L., van der Meulen E., Density-free convergence properties of various estimators of entropy, *Comput. Statist. Data Anal.*, 5:425–436, 1987.
126. Gyorfi L., van der Meulen E., On nonparametric estimation of entropy functionals, in *Nonparametric Functional Estimation and Related Topics*, (G. Roussas, Ed.), Kluwer Academic, Amsterdam, 1990, pp. 81–95.
127. Hampel, F. R., Ronchetti E. M., Rousseau P. J., Stahel W. A., *Robust Statistics: The Approach Based on Influence Functions*. Wiley, New York, 1985.
128. Han S., Rao S., Erdogmus D., Principe J., A minimum error entropy algorithm with self adjusting stepsize (MEE-SAS), *Signal Process.* 87:2733–2745.
129. Han S., Principe J., A Fixed-point Minimum Error Entropy Algorithm, in *Proc. IEEE Int. Workshop on Machine Learning for Signal Processing*, Maynooth, Ireland, 2006.
130. Han S., Rao S., Jeong K., Principe J., A normalized minimum error entropy stochastic algorithm”, *Proc. IEEE Int. Conf. Acoustics Speech and Signal Processing*, Toulouse, 2006.
131. Han S., A Family of Minimum Renyi’s Error Entropy Algorithm for Information Processing, Ph.D. dissertation, University of Florida, Summer, 2007.
132. Han T., Verdu S., Generalizing the Fano inequality, *IEEE Trans. Inf. Theor.*, 40(4):1247–1251, 1994.
133. Hardle W., *Applied Nonparametric Regression*, Econometric Society Monographs vol 19, Cambridge University Press, New York, 1990.
134. Hardoon D., Szekely S., Shawe-Taylor J., Canonical correlation analysis: an overview with application to learning methods, *Neur. Comput.*, 16(12):2664–2699, Dec. 2004.
135. Hart, P., Moment distributions in economics: an exposition, *J. Royal. Statist. Soc. Ser. A*, 138:423–434, 1975.
136. Hartigan J., *Clustering Algorithms*. John Wiley & Sons, New York, 1975.
137. Hartley R., Transmission of information, *Bell Syst. Tech. J.*, 7:535, 1928.
138. Havrda J., Charvat, F., Quantification methods of classification processes: concept of structural a entropy, *Kybernetika* 3:30, 1967.
139. Haykin S. (ed.), *Blind Deconvolution*, Prentice-Hall, Upper Saddle River, NJ, 1994.
140. Haykin S., Principe J., Dynamic modeling with neural networks, in *IEEE Signal Process. Mag.*, 15(3):66–72, 1998.
141. Haykin S., *Neural Networks: A Comprehensive Foundation*, Prentice Hall, Upper Saddle River, NJ, 1999.
142. Haykin S., Sandberg I., Wan E., Principe J., Fancourt C., Katagiri S., *Nonlinear Dynamical Systems: Feedforward Neural Network Perspectives*, John Wiley, New York, 2001
143. Haykin S., *Adaptive Filter Theory*, 4th Edition, Prentice Hall, Englewood Cliffs, NJ, 2002.
144. Hebb D., *Organization of Behavior: A Neurophysiology Theory*, John Wiley, NY, New York, 1949.
145. Hertz J., Krogh A., and Palmer R., *Introduction to the Theory of Neural Computation*, Addison Wesley, Readings, MA, 1991.
146. Heskes T., Energy functions for self-organizing maps. In E. Oja and S. Kaski, editors, *Kohonen Maps*, Elsevier, Amsterdam, 1999, pp. 303–316.

147. Hess W., *Pitch Determination of Speech Signals*. Springer, New York, 1993.
148. Hild II K., Erdogmus D., Principe J., Blind source separation using Renyi's mutual information, *IEEE Signal Process. Lett.*, 8:174–176, 2001.
149. Hild II K., Blind source separation of convolutive mixtures using Renyi's divergence, University of Florida, Gainesville, Fall 2003.
150. Hild II K., Erdogmus D., Principe J., An analysis of entropy estimators for blind source separation, *Signal Process.*, 86(1):182–194, 2006.
151. Hild II K., Erdogmus D., Torkkola K., and Principe J., Feature extraction using information-theoretic learning, *IEEE Trans. Pat. Anal. Mach. Intell.* 28(9):1385–1392, 2006.
152. Hinton G. and Sejnowski T., Unsupervised learning: Foundations of neural computation, MIT Press, Cambridge, MA, 1999.
153. Hofmann T. and Buhmann J., Pairwise Data Clustering by Deterministic Annealing, *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(1):1–14, 1997.
154. Horn D., Gottlieb A., Algorithm for data clustering in pattern recognition problems based on quantum mechanics, *Phys. Rev. Lett.*, 88(1):018702, 2002.
155. Hyvärinen A., Fast and Robust Fixed-Point Algorithms for Independent Component Analysis, *IEEE Trans. Neural Netw.*, 10(3):626–634, 1999.
156. Hyvärinen A., Karhunen J., Oja E., *Independent Component Analysis*, Wiley, New York, 2001.
157. Huber, P.J., Robust Estimation of a Location Parameter. *Ann. Math. Statist.*, 35:73–101, 1964.
158. Huber P., *Robust Statistics*, John Wiley, New York, 1981.
159. Ivanov A., Rozhkova A., Properties of the statistical estimate of the entropy of a random vector with a probability density, *Prob. Inf. Transmiss.*, 17:171–178, 1981.
160. Jain K. and Dubes R., *Algorithms for Clustering Data*. Prentice-Hall, Englewood Cliffs, NJ, 1988.
161. Jaynes E., *Probability Theory, the Logic of Science*, Cambridge University Press, Cambridge, UK, 2003.
162. Jenssen R., Principe J., Erdogmus D. and Eltoft T., The Cauchy–Schwartz divergence and Parzen windowing: Connections to graph theory and mercer kernels, *J. Franklin Inst.*, 343:614–629, 2004.
163. Jenssen R., Erdogmus D., Hild II K., Principe J., Eltoft T., Optimizing the Cauchy–Schwarz PDF divergence for information theoretic, non-parametric clustering, in *Proc. Int'l. Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR2005)*, pp. 34–45, St. Augustine, FL, November 2005.
164. Jenssen R., Erdogmus D., Principe J., Eltoft T., Some equivalence between kernel and information theoretic methods, in *J. VLSI Signal Process.*, 45:49–65, 2006.
165. Jenssen R., Erdogmus D., Hild II K., Principe J., Eltoft T., Information cut for clustering using a gradient descent approach, *Pattern Recogn.*, 40:796–806, 2006.
166. Jeong K.H., Liu W., Principe J., The correntropy MACE filter, *Pattern Recogn.*, 42(5):871–885, 2009.
167. Jeong K.W., Principe J., Enhancing the correntropy MACE filter with random projections, *Neurocomputing*, 72(1–3):102–111, 2008.
168. Jizba P., Toshihiko T., The world according to Renyi: Thermodynamics of multifractal systems, *Ann. Phys.*, 312:17–59, 2004.

169. Joe H., Relative entropy measures of multivariate dependence. *J. Amer. Statist. Assoc.*, 84(405):157–164, 1989.
170. Jones M., McKay I., Hu T., Variable location and scale density estimation, *Ann. Inst. Statist. Math.*, 46:345–52, 1994.
171. Jumarie G., *Relative Information*, Springer Verlag, New York, 1990
172. Jutten C., Herault J., Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture. *Signal Process.*, 24:1–10, 1991.
173. Kailath T., RKHS approach to detection and estimation problems—part I: Deterministic signals in Gaussian noise, *IEEE Trans. Inf. Theor.*, IT-17(5):530–549, Sept. 1971.
174. Kailath T. and Duttweiler D., An RKHS approach to detection and estimation problems—part III: Generalized innovations representations and a likelihood-ratio formula, *IEEE Trans. Inf. Theor.*, IT-18(6):30–45, November 1972.
175. Kailath T. and Weinert H., An RKHS approach to detection and estimation problems—part II: Gaussian signal detection, *IEEE Trans. Inf. Theor.*, IT-21(1):15–23, January 1975.
176. Kalman R., A new approach to linear filtering and prediction problems, *Trans. ASME J. Basic Eng.*, 82:35–45, 1960.
177. Kapur J., *Measures of Information and their Applications*, Wiley Eastern Ltd, New Delhi, 1994.
178. Kass R. and Vos P., *Geometrical Foundations of Asymptotic Inference*, Wiley, New York, 1997.
179. Kawai A., Fukushige T., \$105/Gflops astrophysical N-body simulation with reconfigurable add-in card and hierarchical tree algorithm, in *Proc. SC2006*, IEEE Computer Society Press, Tampa FL, 2006.
180. Kay S., *Modern Spectral Estimation*, Prentice Hall, Englewood Cliffs, NJ, 1988.
181. Kegl B., Krzyzak A., Piecewise linear skeletonization using principal curves. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(1):59–74, 2002.
182. King S., Step-wise clustering procedures. *J. Amer. Statist. Assoc.*, pp. 86–101, 1967.
183. Kohonen T., *Self-Organizing Maps*, 2nd edition, Springer Verlag, New York, 1997.
184. Kolmogorov A., Sur la notion de la moyenne, *Atti della R. Accademia Nazionale dei Lincei*, 12:388–391, 1930.
185. Kolmogorov A., Interpolation and extrapolation of stationary random processes, Rand Co. (translation from the Russian), Santa Monica, CA, 1962.
186. Koontz W., Narendra P., Fukunaga K., A graph theoretic approach to non-parametric cluster analysis, *IEEE Trans. Comput.*, 25:936–944, 1975.
187. Kozachenko L., Leonenko N., Sample estimate of entropy of a random vector, *Prob. Inf. Transmission*, 23:95–101, 1987.
188. Kullback S., *Information theory and statistics*, Dover, Mineola, NY, 1959.
189. Kumar B., Minimum variance synthetic discriminant functions, *J. Opt. Soc. Am., A* 3(10):1579–1584, 1986.
190. Kumar B., Tutorial survey of composite filter designs for optical correlators, *Appl. Opt.*, 31:4773–4801, 1992.
191. Kumar B., Savvides M., Xie C., Venkataramani K., Biometric verification with correlation filters, *Appl. Opt.*, 43(2):391–402, 2004.
192. Kushner H., Yin G., Stochastic approximation and recursive algorithms and applications, *Application of Mathematics* series, vol. 35, Springer, New York, 2003.

193. Lazo A., Rathie P., On the entropy of continuous probability distributions, *IEEE Trans. Inf. Theor.*, 24:120–122, 1978.
194. LeCun Y., Jackel L., Bottou L., Brunot A., Cortes C., Denker J., Drucker H., Guyon I., Muller U., Sackinger E., Simard P., Vapnik V., Learning algorithms for classification: A comparison on handwritten digit reconstruction. *Neural Netw.*, pp. 261–276, 1995.
195. LeCun Y., Bottou L., Bengio Y., Haffner P., Gradient-based learning applied to document recognition, *Proc. IEEE*, 86(11):2278–2324, Nov. 1998.
196. LeCun Y., Chopra S., Hadsell R., Ranzato M., Huang F., A tutorial on energy-based learning, in *Predicting Structured Data*, Bakir, Hofman, Scholkopf, Smola, Taskar (Eds.), MIT Press, Boston, 2006.
197. Lehn-Schieler T., Hegde H., Erdogmus D., and Principe J., Vector-quantization using information theoretic concepts. *Natural Comput.*, 4:39–51, Jan. 2005.
198. Li R., Liu W., Principe J., A unifying criterion for blind source separation based on correntropy, *Signal Process., Special Issue on ICA*, 8(78):1872–1881.
199. Linsker R., Towards an organizing principle for a layered perceptual network. In D. Z. Anderson (Ed.), *Neural Information Processing Systems - Natural and Synthetic*. American Institute of Physics, New York, 1988.
200. Liu W., Pokharel P., Principe J., Error entropy, correntropy and M-estimation, *IEEE Int. Workshop on Machine Learning for Signal Processing*, 2006.
201. Liu W., Pokharel P., Principe J., Correntropy: Properties and applications in non Gaussian signal processing, *IEEE Trans. Sig. Proc.*, 55(11):5286–5298, 2007.
202. Liu W., Pokharel P., Principe J., The kernel LMS algorithm, *IEEE Trans. Signal Process.*, 56(2):543–554, Feb. 2008.
203. Loève, M.M., *Probability Theory*, VanNostrand, Princeton, NJ, 1955.
204. Lorenz E., Deterministic non-periodic flow, *J. Atmospheric Sci.*, 20:130–141, 1963.
205. Lutwak E., Yang D., Zhang G., Cramér–Rao and moment-entropy inequalities for Renyi entropy and generalized Fisher information, *IEEE Trans. Info. Theor.*, 51(2):473–479, 2005.
206. Lyapunov A. *Stability of motion*, Academic Press, New York, 1966
207. MacKay D., *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, Cambridge, UK, 2003.
208. MacQueen J., Some Methods for Classification and Analysis of Multivariate Observations, in *Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.
209. Mahalanobis A., Kumar B., Casasent D., Minimum average correlation energy filters, *Appl. Opt.*, 26(17):3633–3640, 1987.
210. Mahalanobis A., Forman A., Bower M., Cherry R., Day N., Multi-class SAR ATR using shift invariant correlation filters, *Pattern Recogn.*, 27:619–626 *Special Issue on Correlation Filters and Neural Networks*, 1994.
211. Mardia K., Jupp P., *Directional Statistics*, Wiley, New York, 2000.
212. Mari D., Kotz S., *Correlation and Dependence*. Imperial College Press, London, 2001.
213. Marossero D., Erdogmus D., Euliano N., Principe J., Hild II, K., Independent components analysis for fetal electrocardiogram extraction: A case for the data efficient mermaid algorithm, *Proceedings of NNSP'03*, pp. 399–408, Toulouse, France, Sep 2003.

214. Mate L., *Hilbert Space Methods in Science and Engineering*, Hilger, New York, 1989.
215. McCulloch J., Financial applications of stable distributions. In G. S. Madala and C.R. Rao (Eds.), *Handbook of Statistics*, vol. 14, pages 393–425. Elsevier, Amsterdam, 1996.
216. Menendez M., Morales D., Pardo L., Salicru M., Asymptotic behavior and stastistical applications of divergence measures in multinomial populations: a unified study, *Statistical Papers*, 36:129, 1995.
217. Mercer J., Functions of positive and negative type, and their connection with the theory of integral equations, *Philosoph. Trans. Roy. Soc. Lond.*, 209:415–446, 1909.
218. Micheas A. and Zografos K., Measuring stochastic dependence using  $\gamma$ -divergence. *J. Multivar. Anal.*, 97:765–784, 2006.
219. Mika S., Ratsch G., Weston J., Scholkopf B., Muller K., Fisher discriminant analysis with kernels. In *Proceedings of IEEE International Workshop on Neural Networks for Signal Processing*, pages 41–48, Madison, USA, August 23–25, 1999.
220. Middleton D., Statistical-physical models of electromagnetic interference, *IEEE Trans. Electromagn. Compat.*, EMC-19(3):106–126, Aug. 1977.
221. Moller M., A scaled conjugate gradient algorithm for fast supervised learning, *Neural Netw.*, 6:525–533, 1993.
222. Moore E., On properly positive Hermitian matrices, *Bull. Amer. Math. Soc.*, 23(59):66–67, 1916.
223. Morejon R., An information theoretic approach to sonar automatic target recognition, Ph.D. dissertation, University of Florida, Spring 2003
224. Morejon R., Principe J., Advanced parameter search algorithms for information-theoretic learning, *IEEE Trans. Neural Netw. Special Issue Inf. Theor. Learn.*, 15(4):874–884, 2004.
225. Muller K., Smola A., Ratsch G., Scholkopf B., Kohlmorgen J., Vapnik V., Predicting time series with support vector machines. In *Proceedings of International Conference on Artificial Neural Networks*, Lecture Notes in Computer Science, volume 1327, pages 999–1004, Springer-Verlag, Berlin, 1997.
226. Muller K., Mika S., Ratsch G., Tsuda K., Scholkopf B., An introduction to kernel-based learning algorithms. *IEEE Trans. Neural Netw.*, 12(2):181–201, 2001.
227. Murphy P., Ada D., UCI repository of machine learning databases, Tech. Rep., Department of Computational Science, University of California, Irvine, California, USA, 1994.
228. Nadal J., Parga N., Nonlinear neurons in the low noise limit: a factorial code maximizes information transfer, *Network*, 5:561–581, 1994.
229. Nagumo M., Uber eine klasse von mittelwerte, *Japanese J. Math.*, 7:71, 1930.
230. Ng Y., Jordan M., Weiss Y., On spectral clustering: Analysis and an algorithm, in *Advances in Neural Information Processing Systems*, 14, 2001, vol. 2, pp. 849–856.
231. Nikias C. Shao M., *Signal Processing with Alpha-Stable Distributions and Applications*. John Wiley and Sons, New York, 1995.
232. Nilsson N., *Learning Machines*, Morgan Kauffman, San Mateo, Ca, 1933.
233. Oja E., A simplified neuron model as a principal component analyzer. *J. Math. Biol.*, 15:267–273, 1982.

234. Paiva, A.R.C., Park, I., Principe, J.C. A reproducing kernel Hilbert space framework for spike train signal processing. *Neural Comput.*, 21(2):424–449, 2009.
235. Papoulis A., *Probability, Random Variables and Stochastic Processes*, McGraw-Hill, New York, 1965.
236. Pardo L., *Statistical Inference based on Divergence measures*, Chapman & Hall, Boca raton, FL, 2006.
237. Park B., Marron J., Comparison of data-driven bandwidth selectors, *J. Amer. Statist. Assoc.*, 85:66–72, 1990.
238. Parzen E., Statistical inference on time series by Hilbert space methods, *Tech. Report 23, Stat. Dept., Stanford Univ.*, 1959.
239. Parzen E., An approach to time series analysis, *Ann. Math. Stat.*, 32(4):951–989, Dec. 1961.
240. Parzen E., Extraction and detection problems and reproducing kernel Hilbert spaces, *SIAM J. Control*, 1:35–62, 1962
241. Parzen E., On the estimation of a probability density function and the mode, *Ann. Math. Statist.*, 33:1065–1067, 1962.
242. Patterson R., Holdsworth J., Nimmo-Smith I., Rice P., SVOS final report, Part B: Implementing a gammatone filterbank, *Appl. Psychol. Unit Rep.* 2341, 1988
243. Pavlidis T., *Structural Pattern Recognition*. Springer-Verlag, New York, 1977.
244. Pei S., Tseng C., Least mean p-power error criterion for adaptive FIR filter, *IEEE J. Selected Areas Commun.*, 12(9):1540–1547, 1994.
245. Pereira F., Tishby N., Lee L., Distributional clustering of english words. In *Meeting of the Association for Computational Linguistics*, pp. 183–190, 1993.
246. Pham D., Vrins, F., Verleysen, M., On the risk of using Renyi's entropy for blind source separation, *IEEE Trans. Signal Process.*, 56(10):4611–4620, 2008.
247. Plumbley M. Information-theoretic approaches to neural network learning. In A. Browne, editor, *Neural Network Perspectives on Cognition and Adaptive Robotics*, Chapter 4, pp. 72–90. Institute of Physics, Bristol, UK, 1997.
248. Pokharel P., Xu J., Erdogmus D., Principe J., A closed form solution for a nonlinear Wiener filter, *Proc. IEEE Int. Conf. Acoustics Speech and Signal Processing*, Toulouse, France, 2006.
249. Pokharel P., Liu W., Principe J., A low complexity robust detector in impulsive noise, *Signal Process.*, 89(10):1902–1909, 2009.
250. Prichard, D., Theiler, J. (1994). Generating surrogate data for time series with several simultaneously measured variables. *Phys. Rev. Lett.*, 73(7):951–954.
251. Principe J., Xu D., Information theoretic learning using Renyi's quadratic entropy, in *Proc. ICA'99*, 407–412, Aussois, France, 1999.
252. Principe, J., Xu D., Fisher J., Information theoretic learning, in unsupervised adaptive filtering, Simon Haykin (Ed.), pp. 265–319, Wiley, New York, 2000.
253. Principe J., Euliano N., Lefebvre C., *Neural Systems: Fundamentals through Simulations*, CD-ROM textbook, John Wiley, New York, 2000.
254. Principe J., Xu D., Zhao Q., Fisher J. Learning from examples with information theoretic criteria, *VLSI Signal Process. Syst.*, 26:61–77, 2001.
255. Proakis J., *Digital Communications*, Prentice-Hall, Englewood Clifts, NJ, 1988.
256. Ramanan D., Forsyth D., Finding and tracking people from the bottom up, in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, June 2003, pp. 467–474.
257. Ramsay, J., Silverman, B., *Functional Data Analysis*. Springer-Verlag, New York, 1997.

258. Rao Y., D. Erdogmus, G.Y. Rao, J.C. Principe, Fast error whitening algorithms for system identification and control with noisy data, *Neurocomputing*, 69:158–181, 2006.
259. Rao S., Unsupervised Learning: An Information Theoretic Learning Approach, Ph.D. thesis, University of Florida, Gainesville, 2008.
260. Rao, S., Martins A., Principe J., Mean shift: An information theoretic perspective, *Pattern Recogn. Lett.*, 30(1, 3):222–230, 2009.
261. Rao M., Xu J., Seth S., Chen Y., Tagare M., Principe J., Correntropy dependence measure, submitted to *IEEE Trans. Signal Processing*.
262. Renyi A., On measures of dependence. *Acta Mathematica Academiae Scientiarum Hungaricae*, 10:441–451, 1959.
263. Renyi A., On measures of entropy and information, *Proc. of the 4th Berkeley Symp. Math. Statist. Prob. 1960*, vol. I, Berkeley University Press, pp. 457, 1961.
264. Renyi A., Probability Theory, North-Holland, University Amsterdam, 1970.
265. Renyi A. (Ed.), *Selected Papers of Alfred Renyi*, vol. 2, Akademia Kiado, Budapest, 1976.
266. Renyi A., Some fundamental questions about information theory, in Renyi, A. (Ed.), *Selected Papers of Alfred Renyi*, vol. 2, Akademia Kiado, Budapest, 1976.
267. Riedmiller M., Braun H., A direct adaptive method for faster backpropagation learning: The RPROP Algorithm, in *Proc. of the IEEE Intl. Conf. on Neural Networks*, San Francisco, 1993, pp. 586–591.
268. Ripley B., *Pattern Recognition and Neural Networks*, Cambridge University Press, New York, 1996
269. Roberts S., Everson R., Rezek I., Maximum certainty data partitioning, *Pattern Recogn.*, 33:833–839, 2000.
270. Roberts S., Holmes C., Denison D., Minimum entropy data partitioning using reversible jump Markov chain Monte Carlo, *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(8):909–914, 2001.
271. Rose K., Gurewitz E., Fox G., Vector quantization by deterministic annealing, *IEEE Trans. Inf. Theor.*, 38(4):1249–1257, 1992.
272. Rosenblatt M., Remarks on some nonparametric estimates of a density function, *Ann. Math. Statist.*, 27:832–837, 1956.
273. Ross T., Worrell S., Velten V., Mossing J., Bryant M., Standard SAR ATR evaluation experiments using the MSTAR public release data set, in: *Proceedings of the SPIE*, vol. 3370, 1998, pp. 566–573.
274. Roweis S., Saul L., Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.
275. Rubinstein R., *Simulation and the Monte Carlo Method*, John Wiley & Sons, New York, 1981.
276. Rudin W. *Principles of Mathematical Analysis*. McGraw-Hill, New York, 1976.
277. Rumelhart D., McClelland J., Eds, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1, MIT Press, Cambridge, MA, 1986.
278. Salicru M., Menendez M., Morales D., Pardo L., Asymptotic distribution of  $(h,\phi)$ -entropies, *Comm. Statist. Theor. Meth.*, 22(7):2015–2031, 1993.
279. Sands N., Cioffi J., Nonlinear channel models for digital magnetic recording, *IEEE Trans. Magnetics*, 29(6):3996–3998, Nov 1993.

280. Sands N., Cioffi J., An improved detector for channels with nonlinear intersymbol interference, *Proc. Intl. Conf. on Communications*, vol 2, pp 1226–1230, 1994.
281. Santamaría I., Erdogmus D., Principe J.C., Entropy minimization for supervised communication channel equalization, *IEEE Trans. Signal Process.*, 50(5):1184–1192, 2002.
282. Santamaría I., Pokharel P., Principe J., Generalized correlation function: Definition, properties and application to blind equalization, *IEEE Trans. Signal Process.*, 54(6):2187–2197, 2006.
283. Santos J., Alexandre L., Sa J., The error entropy minimization algorithm for neural network classification, in A. Lofti (Ed.), *Int Conf. Recent Advances in Soft Computing*, pp. 92–97, 2004.
284. Sayed A., *Fundamentals of Adaptive Filters*, John Wiley & Son, New York, 2003
285. Sheppard A., *Second Order Methods for Neural Networks*, Springer, London, 1997.
286. Scanlon J., Deo N., Graph-theoretic algorithms for image segmentation. In *IEEE International Symposium on Circuits and Systems*, pp. VI141–144, Orlando, Florida, 1999.
287. Schölkopf B., Smola A., Muller K., Nonlinear component analysis as a kernel eigenvalue problem, *Neural Comput.*, 10:1299–1319, 1998.
288. Schölkopf, B., Burges, C.J.C., Smola, A.J. (Eds.), *Advances in Kernel Methods: Support Vector Learning*. MIT Press, Cambridge, MA, 1999.
289. Schölkopf B. and Smola A., *Learning with Kernels*. MIT Press, Cambridge, MA, 2002
290. Schrauwen, B., Campenhout, J.V., Linking non-binned spike train kernels to several existing spike train distances. *Neurocomp.* 70(7–8), 1247–1253 (2007).
291. Schreiber, T., Schmitz, A. (2000). Surrogate time series. *Physica D*, 142:346–382.
292. Seth S., and Principe J., On speeding up computation in information theoretic learning, in *Proc. IJCNN 2009*, Atlanta, GA, 2009.
293. Shannon C., and Weaver W., *The mathematical Theory of Communication*, University of Illinois Press, Urbana, 1949.
294. Shawe-Taylor J. Cristianini N., *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, UK, 2004.
295. Sheather S. Jones M., A reliable data-based bandwidth selection method for kernel density estimation, *J. Roy. Statist. Soc., Ser. B*, 53:683–690, 1991.
296. Shi J., Malik J., Normalized cuts and image segmentation, *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.
297. Sidak Z., Sen P., Hajek J., *Theory of Rank Tests*, Academic Press, London, 1999.
298. Silva L., Felgueiras C., Alexandre L., Sa J., Error entropy in classification problems: a univariate data analysis, *Neural comput.*, 18(9):2036–2061, 2006.
299. Silva L., *Neural networks with error density risk functionals for data classification*, Ph.D. Thesis, Faculdade de Engenharia, University of Porto, 2008.
300. Silverman B., *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, London, 1986.
301. Silvey S., On a measure of association. *Ann Math Statist.*, 35(3): 1157–1166, 1964.
302. Singh A., Principe J., Using correntropy as a cost function in linear adaptive filters, *Proc. IEEE IJCNN 2009*, Atlanta, 2009.

303. Singh A. and Principe J., "Information Theoretic Learning with Adaptive Kernels", *Signal Processing*, 2009 (submitted).
304. Slaney M., Malcolm Slaney's Auditory Toolbox to implement auditory models and generate synthetic vowels. <http://www.slaney.org/malcolm/pubs.html>.
305. Slonim N. Tishby N., The power of word clusters for text classification. In *23rd European Colloquium on Information Retrieval Research*, 2001.
306. Sneath P. Sokal R., *Numerical Taxonomy*. Freeman, London, 1973.
307. Snyder, D.L., *Random Point Process in Time and Space*. John Wiley & Sons, New York, 1975.
308. Song, K., Renyi information, log likelihood and an intrinsic distribution measure, *J. of Stat. Plan. and Inference*, 93: 51–69, 2001.
309. Stoller D., Univariate two population distribution free discrimination, *J. Amer. Stat. Assoc.* 49: 770–777, 1954.
310. Styblinski M., Tang T., Experiments in nonconvex optimization: Stochastic approximation with function smoothing and simulated annealing, *Neural Netw.*, 3: 467–483, 1990.
311. Suykens J., Gestel T., Brabanter J., Moor B., Vandewalle J., Least Squares Support Vector Machines, *Word Scientific*, Singapore, 2002.
312. Takens F., On the numerical determination of the dimension of an attractor, in Rand and Young, Eds, *Dynamical systems and turbulence*, A. Notes in Mathematics, vol. 898, pp. 366–381, Springer Verlag, Berlin, 1981.
313. Tanrikulu O., Chambers J., Convergence and steady-state properties of the least-mean mixed norm (LMMN) adaptive algorithm, *IEE Proc. -Vision, Image Signal Process.*, 143: 137–142, June 1996.
314. Terrel G., Scott D., Variable kernel density estimation, *Ann. Statist.* 20: 1236–65, 1992.
315. Theodoridis S., K. Koutroumbas, *Pattern Recognition*, Academic Press, 1999.
316. Tishby N., Pereira F., and Bialek W., The information bottleneck method. In *Proceedings of the 37th Annual Allerton Conference on Communication, Control and Computing*, pp. 368–377, 1999.
317. Tishby N., Slonim N., Data clustering by markovian relaxation and the information bottleneck method, in *Advances in Neural Information Processing Systems*, 13, Denver, pp. 640–646, 2000.
318. Torkkola K., Visualizing class structure in data using mutual information, *Proceedings of NNNSP X*, pp. 376–385, Sydney, Australia, 2000
319. Torkkola K., Feature extraction by non-parametric mutual information maximization, *J. Mach. Learn. Res.*, 3:1415–1438, 2003.
320. Tsallis C., Possible generalization of Boltzmann Gibbs statistics, *J. Stat. Phys.*, 52:479, 1988.
321. Tsybakov A., van der Meulen E., Root-n consistent estimators of entropy for densities with unbounded support, *Scand. J. Statist.*, 23:75–83, 1994.
322. Urquhart R., Graph theoretical clustering based on limited neighbor sets, *Pattern Recogn.*, 173–187, 1982
323. Vapnik V., *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, 1995
324. Velten V., Ross T., Mossing J., Worrell S., Bryant M., Standard SAR ATR evaluation experiments using the MSTAR public release data set, research report, Wright State University, 1998.
325. Viola P., Schraudolph N., Sejnowski T., Empirical entropy manipulation for real-world problems, *Proc. NIPS'95*, pp. 851–857, 1995

326. von Neumann, J., *Mathematical Foundations of Quantum Mechanics*, Princeton University Press, Princeton, NJ, 1955.
327. Wahba G., *Spline Models for Observational Data*, SIAM, . Philadelphia, PA, 1990, vol. 49.
328. Walach E., Widrow B., The least mean fourth (LMF) adaptive algorithm and its family, *IEEE Trans. Inf. Theor.*, IT-30(2):275–283, 1984.
329. Wang D., Brown G., *Computational Auditory Scene Analysis—Principles, Algorithms, and Applications*. Wiley, New York, 2006.
330. Watanabe S., *Pattern Recognition: Human and Mechanical*. Wiley, New York, 1985.
331. Werbos P., *Beyond regression: New tools for prediction and analysis in the behavioral sciences*, Ph.D. Thesis, Harvard University, Cambridge, 1974.
332. Widrow B., S. Stearns, *Adaptive Signal Processing*, Prentice Hall, Englewood Cliffs, NJ, 1985.
333. Wiener N., *Nonlinear Problems in Random Theory*, MIT, Boston, 1958.
334. Wittner B. Denker J., Strategies for teaching layered networks classification tasks, in *Neural Inf. Proc. Syst.* (Ed Anderson), 850–859, Ame. Inst. Phys. 1987.
335. Wu H., Principe J., Simultaneous diagonalization in the frequency domain for source separation, *Proc. First Int. Workshop on Ind. Comp. Anal. ICA'99*, 245–250, Aussois, France, 1999.
336. Wu Z. and Leahy R., An optimal graph theoretic approach to data clustering: Theory and its applications to image segmentation. *IEEE Trans. Pattern Anal. and Mach. Intell.*, 15(11):1101–1113, 1993.
337. Wyszecki G., Stiles W., *Color Science: Concepts and Methods, Quantitative Data and Formulae*. Wiley, New York, 1982.
338. Xie C., Savvides M., Kumar B., Kernel correlation filter based redundant class dependence feature analysis on FRGC2.0 data, in: *Proc. second Int. Workshop Analysis Modeling Faces Gesture (AMFG)*, Beijing, 2005.
339. Xu D., Principe J., Fisher J., Wu H., A novel measure for independent component analysis (ICA), in *Proc. of ICASSP'98*, vol. 2, pp. 1161–1164, 1998
340. Xu D., Energy, Entropy and Information Potential for Neural Computation, PhD Dissertation, University of Florida, Gainesville, 1999
341. Xu J., Pokharel P., Jeong K., Principe J., An explicit construction of a reproducing Gaussian kernel Hilbert space, *Proc. IEEE Int. Conf. Acoustics Speech and Signal Processing*, Toulouse, France, 2005.
342. Xu J., Principe J., A pitch detector based on a generalized correlation function, *IEEE Trans. Audio, Speech Lang. Process.*, 16(8):1420–1432, 2008.
343. Xu J., *Nonlinear Signal Processing Based on Reproducing Kernel Hilbert Space*, Ph.D. Thesis, University of Florida, Gainesville, 2008.
344. Xu J., Bakardjian H., Cichocki A., Principe J., A new nonlinear similarity measure for multichannel signals, *Neural Netw.* (invited paper), 21(2–3):222–231, 2008.
345. Yang C., Duraiswami R., Gumerov N., Davis L., Improved fast Gauss transform and efficient kernel density estimation. In *Proc. ICCV 2003*, pages 464–471, 2003.
346. Yang H., Amari S., Adaptive online learning algorithms for blind separation: maximum entropy and minimum mutual information, *Neural Comput.*, 9:1457–1482, 1997

347. Zahn T., Graph theoretic methods for detecting and describing gestalt clusters. *IEEE Trans. Comput.*, 20:68–86, 1971.
348. Zemel R., Hinton G., Learning population codes by minimizing the description length, in *Unsupervised Learning*, Hinton and Sejnowski (Eds.), pp. 261–276, MIT Press, Cambridge, MA, 1999.
349. Zhao Q., Principe J., Brennan V., Xu D., Wang Z., Synthetic aperture radar automatic target recognition with three strategies of learning and representation, *Opt. Eng.*, 39(5):1230–1244, 2000.
350. Zien A., Ratsch G., Mika S., Schölkopf B., Lengauer T., Muller K., Engineering support vector machine kernels that recognize translation invariant sites in DNA. *Bioinformatics*, 16:906–914, 2000.

---

# Index

- ( $h, \phi$ ) divergences, 20  
( $h, \phi$ ) entropies, 19  
 $F$ -correlation function, 366  
 $\alpha$  -stable distributions, 404  
 $\alpha$  information forces, 79  
 $\alpha$  moment of the probability mass function, 54  
 $\alpha$ -Information Potential, 79  
 $\phi$  divergences, 19  
 $\phi$ -entropy, 19  
 $\alpha$  divergence, 20  
 $\alpha$  information potential, 50, 79  
 $\alpha$  information potential estimator, 60  
 $\alpha$ -information potential, 229  
 $\alpha$ -information potential, 52, 147
- acoustic noise cancellation, 174  
adaline, 330  
adaptation of the kernel size, 135, 229  
adapting the kernel size, 216  
adaptive filter in RKHS, 39  
adaptive filtering, 103, 123, 135  
adaptive information filtering, 25  
adaptive kernel, 214  
adaptive kernel size, 190  
adaptive learning, 169  
adaptive noise cancellation, 138  
adaptive stepsize algorithms, 121  
adaptive systems, 24  
advanced search methods, 186  
affinity matrix, 293  
Alfred Renyi, 47  
ambiguous concavity, 51
- AMISE, 294  
amount of information, 10, 48  
analytic solution, 117  
annealed kernel size algorithm, 203  
annealing phase, 113  
annealing procedure, 295  
annealing the kernel, 113, 278  
anti-Hebbian, 306, 312  
arbitrary kernels, 79  
aspect angle, 237  
asymmetric kernel, 281  
asymptotic equipartition property, 12  
asymptotic MISE, 72, 458  
asymptotically unbiased, 56  
attraction force, 229  
attractive force field, 75  
autocorrelation matrix, 26  
autocorrentropy, 429  
autocorrentropy function, 415  
autocorrentropy matrix, 419  
autoencoders, 309  
automatic target detection, 448  
automatic target recognition, 237  
average distance, 268  
axioms, 13, 50
- backpropagation algorithm, 181  
backpropagation learning, 245  
backpropagation of information forces, 184  
bandwidth parameter, 112  
batch size selection, 483  
Bayes-NP (nonparametric), 250

- Bayes-P (parametric), 250  
 Bayesian estimation, 458  
 Bayesian inference, 315  
 Bayesian parameter estimation, 460  
 Bhattacharyya distance, 85, 269  
 bias, 56, 72, 458  
 bias term, 108  
 bias–variance trade-off, 468  
 biased estimate, 467  
 bisquare, 126  
 bisquare weights, 209  
 bit error rate, 208  
 blind deconvolution, 335  
 blind equalization, 33  
 blind source separation (BSS), 310  
 blurring mean shift, 282  
 Bregman divergence, 20  
 canonical correlation, 307  
 Cauchy – Schwarz PDF divergence, 317  
 Cauchy's functional equation, 48  
 Cauchy–Schwarz information forces, 89  
 Cauchy–Schwarz information potential, 89  
 Cauchy–Schwarz quadratic mutual information, 87, 367  
 Cauchy-Schwarz, 233  
 Cauchy-Schwarz divergence, 86, 219, 273, 292, 359  
 Cauchy-Schwarz inequality, 86  
 CDM, 401  
 CEF, 271  
 center of the cluster, 77  
 centered autocorrentropy function, 422  
 centered cross-correntropy, 396  
 centered error entropy criterion, 134  
 centered Gram matrix, 440  
 centering CMACE, 431  
 chain rule, 26, 109, 182  
 changing curvature, 154  
 channel capacity, 299  
 chaotic system, 198  
 Chebyshev entropy, 53  
 Chernoff distance, 85  
 chordal distance, 85  
 CIP, 284  
 CIP kernel, 377  
 class conditional PDFs, 228  
 class divergence, 229  
 classification, 31, 219  
 classification error, 231  
 classification process, 257  
 classifier, 247  
 Classifier trained with MEE, 225  
 cluster compactness, 274  
 clustering, 32, 264  
 clustering evaluation function, 266, 268  
 CMACE, 449  
 CMACE filter, 429  
 CMACE Filter Implementation, 430  
 cochleogram, 433  
 competitive networks, 264  
 complexity–accuracy dilemma, 23  
 computational complexity, 165, 277  
 concave, 51  
 conditional entropy, 15, 51, 231  
 conditional expectation, 28  
 confusion matrix, 254  
 congruence, 42  
 congruence mapping, 353, 356  
 congruent, 427  
 conjugate gradient algorithms, 187  
 consistent in quadratic mean, 56  
 constant modulus, 206  
 constant variance constraint, 336  
 constrained gradient descent, 273  
 constrained optimization, 275  
 constrained optimization problem, 372  
 contrast function, 341, 442  
 convex, 51  
 convolution smoothing, 113, 494  
 convolution smoothing for ITL, 113  
 correlation learning, 329  
 correntropy, 104, 131, 385, 395, 415  
 correntropy coefficient, 397, 409  
 correntropy dependence measure, 401  
 correntropy energy, 429  
 correntropy function, 415  
 correntropy ICA, 447  
 correntropy in Time Series Analysis, 416  
 correntropy induced metric, 128  
 correntropy KLT (CoKLT), 438  
 correntropy MACE (CMACE), 427  
 correntropy matched filter, 402  
 correntropy matched filter statistic, 403  
 correntropy spectral density, 441  
 correntropy spectral density (CSD), 423

- correntropy-gram, 433
- cost function, 26, 225
- cost function for ICA, 444
- covariance kernel, 34, 41
- covariance of a random process, 41
- Cover's theorem, 351
- cross correlation vector, 27
- cross entropy, 16
- cross information potential, 7, 87, 264, 284, 353, 358
- cross-correntropy, 124, 386, 428, 444
- cross-correntropy function, 424
- cross-entropy, 38, 136
- cross-entropy cost function, 222
- cross-information potential, 88, 266, 359
- cross-validation estimate, 473
- curvature analysis, 156
  
- data clusters, 266
- data modeling, 1
- decision feedback equalizer, 204
- deconvolving filter, 339
- deep networks, 243
- delta autocorrelation, 117
- delta cross-correlation, 117
- density estimation, 457
- dependence measure, 400
- dependence on input power, 160
- designing a kernel, 407
- deterministic functionals, 353
- differences of error samples, 109
- differential entropy, 266
- differential entropy clustering, 267
- differential Renyi's entropy, 55
- digital communications, 204
- digital satellite communications, 206
- dilation in the  $e$ -space, 496
- dimensionality reduction, 232
- directed trees, 288
- discover complex mappings, 243
- discriminability-preserving projection, 241
- discriminant function, 220, 228, 264
- discriminative projections, 231
- discriminative subspace, 231
- discriminative training, 224
- dissimilarity, 16, 263, 267
- distance, 266
- divergence, 16, 191, 263
  
- divergence measures, 219
- divergence mutual information, 31
- divergences, 85
- dual Lagrangian, 372
- dynamic modeling, 198
- dynamic performance surface, 191
  
- ECC cost function, 172
- Echo return loss enhancement, 176
- EEC criterion, 181, 339
- EEC minimum, 115
- EEC performance surface, 114
- effect of dimension in QMI estimators, 491
- effect of kernel size, 156
- effect of scale in QMI estimators, 490
- effects of sample size in QMI estimators, 486
  
- eigendecomposition problem, 380
- eigenvalues of a matrix, 120
- embedding space dimension, 199
- entropic criterion, 103
- entropy, 9, 11, 49, 84, 264, 316, 317
- entropy costs, 33
- entropy estimation, 57, 168
- entropy estimator, 69
- entropy minimization, 70, 300
- entropy order, 122
- entropy weighted incremental autocorrelation, 163
- entropy weighted incremental cross-correlation, 163
- Epanechnikov kernel, 469
- error correntropy criterion, 124
- error entropy, 112, 358
- error entropy criterion, 29, 33, 104, 141, 222
- error entropy minimization, 205
- error-whitening LMS, 160
- estimate divergences, 48
- estimate of correntropy coefficient, 398
- estimate of Renyi's order- $\alpha$  divergence, 82
- estimate of Shannon's entropy, 151
- estimate of the joint PDF, 64
- estimated Cauchy-Schwarz mutual information, 92
- estimated Euclidean mutual information, 92

- estimating Shannon entropy, 80  
 Estimator for  $\alpha$ -Renyi's Entropy, 59  
 estimator for center correntropy, 397  
 estimator in Renyi's entropy, 73  
 estimator of the information potential, 361  
 estimators of quadratic distances, 362  
 Euclidean distance between PDFs, 85  
 Euclidean distance measure, 365  
 Euclidean divergence, 233  
 exact recursive entropy estimator, 143  
 expectation maximization (EM) algorithm, 271  
 expectation operator, 363  
 expected value of functionals, 364  
 factor analysis, 311  
 factorial code, 308  
 false positives, 237  
 families of entropy curves, 475  
 Fano bound, 219, 231, 253  
 fast computation of IP, 99  
 fast computation of the CIP, 100  
 fast Gauss transform, 59, 96, 167  
 fastICA, 348  
 feature extraction, 31  
 feature extractor, 247  
 feature ranking, 250  
 feedforward training, 243  
 fiducial point, 134  
 filtering, 31  
 finite impulse response, 141  
 finite impulse response (FIR) filters, 26  
 Fisher information, 111  
 fixed-point algorithm, 117, 162, 318  
 fixed-point equation, 282  
 fixed-point MEE algorithm, 166  
 fixed-point minimum error entropy, 163  
 flat spots, 186  
 FLOPS, 196  
 Fock spaces, 35  
 forgetting factor, 164  
 forgetting recursive entropy estimator, 144  
 forgetting recursive estimator, 164  
 Fourier transform, 425  
 frequency doubler, 195, 244  
 furthest-point clustering, 97  
 fuzzy clustering, 275  
 gain in information, 14  
 gammatone filters, 433  
 Gaussian, 151  
 Gaussian blurring mean shift, 282  
 Gaussian distributed residuals, 123  
 Gaussian function, 38, 97  
 Gaussian kernel, 70, 75, 78, 109, 124, 129, 147, 162, 172, 200, 241, 248, 360, 388  
 Gaussian mean shift, 282  
 Gaussian processes, 34  
 Gaussian residual, 25  
 GBMS, 284, 318  
 general gradient form, 233  
 generalization of maximum likelihood, 126  
 generalized correlation function, 385, 416  
 generalized cross information potential, 89  
 generalized Gaussian density, 347  
 generalized Information forces, 92  
 generalized information potential, 33, 89, 235  
 generalized XOR, 226  
 geodesic distance, 85  
 geometric interpretation Renyi's entropy, 52  
 Given's rotation matrix, 249  
 global convergence theorem, 494  
 global minimum, 113  
 global optimization, 113, 202  
 gradient ascent, 137, 172  
 gradient computation, 182  
 gradient descent, 27  
 gradient for the MEE-RIP, 146  
 gradient of Shannon's entropy, 151  
 gradient vector, 27  
 Gram matrix, 39, 43, 366, 379  
 Gram–Charlier series expansion, 462  
 graph cut, 264, 286  
 graph spectral clustering, 287  
 graph theory, 292  
 graph-theoretic algorithms, 273  
 graph-theoretic clustering, 286  
 greedy search, 270  
 Hadamard product, 170  
 Hartley's entropy, 53

- Havrda-Charvat, 54, 85  
 Hebbian learning, 304, 329  
 Hebbian term, 306  
 Hellinger's distance, 85  
 Hermite interpolation, 168  
 Hermite polynomials, 96, 463  
 Hessian approximation, 153  
 Hessian matrix, 67, 118, 188  
 hierarchical features, 318  
 higher kurtosis, 207  
 Hilbert space, 34  
 Hilbert–Schmidt polynomials, 35  
 histogram density estimation, 464  
 hyperspherical symmetry, 68
- I-Max principle, 307  
 identifying and tracking, 214  
 impulsive noise, 173, 216  
 incomplete Cholesky decomposition, 59, 99, 170  
 incremental errors, 111  
 Independent Component Analysis, 25, 32, 310  
 InfoMax, 329, 348  
 InfoMax principle, 312  
 information bottleneck (IB) method, 23, 300, 313  
 information bottleneck framework, 32  
 information cut, 292, 293  
 information divergence in classification, 231  
 information filter, 243  
 information filtering, 181, 231  
 information filtering with QMI, 234  
 Information Forces, 73–75, 110, 183, 228, 234, 236, 288  
 information maximization, 300  
 information maximization principle (InfoMax), 304  
 information particles, 73, 77, 110, 229  
 information potential, 6, 21, 33, 58, 88, 105, 110, 143, 147, 225, 228, 241, 263, 268, 274, 293, 371, 415  
 information potential (IP) estimator, 47  
 information potential and force in classification, 229  
 information potential estimator, 77  
 information potential field, 73, 88  
 information regression, 25
- information theoretic interpretation, 352  
 Information theory, 7, 8, 22, 29  
 information transfer, 23  
 information-theoretic, 250  
 information-theoretic clustering, 264  
 Information-theoretic descriptors, 299  
 information-theoretic interpretation, 88  
 information-theoretic learning, 5, 6, 29, 352, 357  
 information-theoretic method, 247, 265  
 information-theoretic performance measures, 25  
 information-theoretic perspective, 136  
 information-theoretic principles, 306  
 information-theoretic vector quantization, 327  
 injected error, 110  
 inner product, 358  
 inner product in ITL RKHS, 360  
 inner-product distances, 85  
 inner-product operator, 39  
 integral estimates, 472  
 intensity functions, 376  
 inter cluster compactness, 274  
 intersymbol interference, 204  
 intrinsic shape of the PDF, 50  
 inverse locally weighted incremental autocorrelation, 165  
 IP estimator bias, 70  
 IP estimator variance, 71  
 IP gradient, 110  
 IP of the factorized marginal, 89  
 IP of the joint PDF, 89  
 isometric isomorphism, 42, 356  
 ITL, 231  
 ITL algorithms, 263  
 ITL cost functions, 358  
 ITL fuzzy algorithm, 277  
 ITL fuzzy clustering, 278  
 ITL information forces, 192, 193  
 ITL RKHS, 352, 376, 378  
 ITL training, 192  
 ITL-based principles, 300
- J divergence, 16  
 Jensen's inequality, 69, 81, 255  
 joint diagonalization, 443  
 joint entropy, 15

- joint entropy estimation, 68  
 joint output entropy, 341  
 k-means algorithm, 264  
 Karhunen–Loeve transform, 438  
 kernel (Parzen) estimation, 29, 56  
 kernel annealing, 245, 278  
 kernel bandwidth, 133  
 kernel canonical correlation, 400  
 kernel constrained covariance (COCO), 401  
 kernel correlation filter, 450  
 kernel density estimation, 38, 56, 58, 96, 464, 465  
 kernel density estimators, 29  
 kernel dependence functionals, 400  
 kernel eigenfunctions, 40  
 kernel function, 56, 67, 355  
 kernel ICA, 364  
 kernel methods, 35, 39  
 kernel size, 56, 112, 114, 115, 121, 122, 137, 190, 227, 273, 483  
 kernel size maxima, 479  
 kernel size ranges, 478  
 kernel trick, 35, 351  
 kernel-based clustering, 273  
 kernel-based learning theory, 37  
 KL divergence, 83, 90, 316  
 KL divergence as square distance, 16  
 KLDA, 37  
 KLMS algorithm, 40  
 KPCA, 37  
 Kullback–Leibler (KL) divergence, 16, 136, 302, 459  
 kurtosis, 13, 345, 347  
 $L^*u^*v$  space, 322  
 lag domain, 432  
 Lagrange multiplier, 159, 301, 379  
 Lagrange multiplier formalism, 273  
 Laguerre functions, 463  
 Laplace transform, 495  
 largest eigenvalue, 27  
 LDA, 232  
 learning by kernel annealing, 277  
 least absolute residuals, 211  
 least angle regression, 209  
 least mean square, 24  
 least square solution, 117  
 least squares, 117  
 least-square algorithm, 24  
 Legendre series expansion, 462  
 Levenberg–Marquardt, 189, 194  
 likelihood function, 220, 229  
 likelihood statistic, 407  
 Linde–Buzo–Gray, 327  
 line search algorithm, 193  
 line search technique, 187  
 linear combiner, 148  
 linear convergence rate, 283  
 linear decision boundary, 204  
 linear discriminant analysis, 250  
 linear entropy, 54  
 linear transversal equalizer, 204  
 LMS algorithm, 27, 103, 147, 172  
 local convergence of MEE-FP, 163  
 local minima, 113, 186, 346  
 localized variance of the error, 138  
 logistic function, 313  
 lower and upper bounds, 258  
 M Estimation, 104, 126  
 MACE, 448  
 machine learning, 22  
 Mackey–Glass system, 198  
 macroscopic modeling level, 1  
 Mahalanobis distance, 135  
 manifold learning, 324  
 MAP, 229, 232, 237, 371  
 MAP Classifiers, 368  
 marginal error entropies, 142  
 marginal kernels, 64  
 matched filtering, 402  
 matched spatial filters, 448  
 matrix inversion lemma, 165  
 matrix-vector products, 96  
 MaxEnt, 300  
 maximally informative, 238  
 maximization of output power, 329  
 maximization of Renyi's mutual information (MRMI), 248  
 maximizing divergences, 266  
 maximizing entropy, 77, 332  
 maximizing information potential, 109  
 maximizing mutual information, 32  
 maximum a posteriori (MAP), 220, 460  
 maximum correntropy criterion, 125  
 maximum eigenvalue, 121

- maximum entropy, 11  
 maximum entropy deconvolution, 336  
 maximum entropy principle, 301, 330  
 maximum information transfer, 32  
 maximum likelihood estimation, 458  
 maximum likelihood principle, 228  
 maximum log-likelihood, 137  
 maximum mean discrepancy, 364  
 maximum mutual information, 245  
 maximum variance, 378  
 MCC algorithm, 172, 210  
 MCC-SIG(1), 172  
 mCI kernel, 378  
 mean integrated square error (MISE), 458  
 mean of PDF, 106  
 mean shift algorithm, 264, 282, 283, 318  
 mean square algorithm, 27  
 mean square error, 24, 123, 458  
 mean square error criterion, 141  
 measure of dependence, 400, 401  
 MEE, 199  
 MEE algorithm, 110, 142, 166, 181  
 MEE criterion, 174  
 MEE delta-rule, 183  
 MEE-BP algorithm, 183, 200, 206, 212, 213, 219  
 MEE-FP, 167  
 MEE-RIP, 144  
 MEE-SAS, 157  
 MEE-SAS cost function, 152, 154  
 MEE-self-adjusting stepsize, 152  
 MEE-SIG, 146, 147  
 membership function, 270  
 memoryless cross-intensity (mCI) kernel, 377  
 Mercer kernel, 37, 352  
 Mercer kernel-based learning, 351  
 Mercer learning, 37  
 Mercer theorem, 37, 391  
 mesoscopic modeling level, 2  
 metric, 128, 266  
 MI-based matched filter, 404  
 microscopic modeling level, 2  
 Middleton model, 132, 209  
 MIMO information filter, 232  
 MIN-MAX problem, 23  
 minimization of cross-entropy, 303  
 minimization of divergence, 31  
 minimization of error entropy, 105, 141  
 minimization of mutual information, 341  
 minimization of Renyi's mutual information, 343  
 minimization of variance, 28  
 minimum average correlation energy filter (MACE), 425  
 minimum classification error, 250  
 minimum entropy deconvolution, 336  
 minimum entropy encoding, 309  
 minimum error entropy, 24  
 minimum kernel size, 479  
 minimum mean-square-error, 28  
 minimum probability of error, 220  
 minimum redundancy, 309  
 minimum variance criterion, 264  
 MinKL principle, 303  
 MinXEnt, 300, 303  
 misadjustment, 149, 160, 167  
 misclassification rate, 247  
 MISE, 468  
 missed detections, 237  
 mixture density models, 461  
 mixture of Gaussians, 461  
 ML kernel size estimation, 470  
 ML principle, 459  
 MLP, 271, 332  
 modal matrix, 119  
 model-trust-region, 188  
 moments of the data, 13  
 moments of the distribution, 301  
 momentum term, 186  
 Monte Carlo simulations, 199, 210, 212  
 Moore–Aronszajn theorem, 36  
 MRMI, 347  
 MRMI-SIG, 249, 348  
 MSE, 199, 221  
 MSE cost function, 113, 183  
 MSE criterion, 117, 119, 210  
 MSE of kernel estimates, 468  
 multidimensional Hermite function, 97  
 multidimensional kernel, 64, 68  
 multilayer perceptron, 204, 220  
 multiple system outputs, 142  
 multiple-input–multiple-output, 32, 169  
 multivariate surrogate data, 409  
 multivariate Taylor series, 98  
 mutual information, 9, 14, 15, 48, 231

- nearest neighbor density estimation, 465  
 negentropy, 341  
 neighborhood, 290  
 neural network theory, 181  
 Newton's method, 188  
 NMEE algorithm, 158  
 non-Gaussian, 172, 181  
 non-parametric approach, 41  
 nonlinear adaptive filtering, 213  
 nonlinear couplings, 409  
 nonlinear demixing networks, 342  
 nonlinear equalizer, 204  
 nonlinear ICA, 32  
 nonlinear regression, 212  
 nonlinear topology, 182  
 nonlinear weighting function, 268  
 nonparametric regression, 136  
 nonparametric, 457  
 nonparametric classifiers, 228  
 nonparametric density estimation, 464  
 nonparametric estimator, 79  
 normal equations, 44  
 normalized cut, 287, 294  
 normalized information potential, 115, 131  
 normalized LMS algorithm, 158  
 normalized MEE, 157  
 novelty filtering, 33  
  
 object recognition, 448  
 Occam's razor principle, 23  
 on-line update, 265  
 one-step secant, 189  
 online adaptation, 213  
 open set classification, 237, 242  
 optimal Filtering, 27, 43  
 optimal least square solution, 41  
 optimal weight, 114  
 ordered derivative, 182  
 orthogonality condition, 28  
 orthonormal vectors, 378  
 outliers, 132  
 output joint entropy, 311  
  
 parameter optimization, 186  
 parametric and nonparametric model building, 3  
 parametric Bayesian, 457  
  
 parametric centered correntropy, 399  
 parametric correntropy, 393, 394  
 parametric correntropy coefficient, 399  
 parametric density estimation, 458  
 Park and Marron, 469  
 partition function, 314  
 partitioning method, 286  
 Parzen estimation, 72  
 Parzen estimators, 84, 229, 370  
 Parzen method, 388  
 Parzen PDF estimate, 267, 279  
 Parzen window, 237, 352, 465  
 Parzen window size, 294  
 PCA, 232, 438  
 PCA of spike trains, 378  
 performance assessment index, 189  
 performance surface, 27, 121, 122, 186  
 physical entropy, 51  
 pitch determination algorithm, 433  
 plug-in estimates, 472  
 PMF  $\alpha$ -norm, 52  
 point processes, 376  
 postclustering, 291  
 posterior probability, 220  
 posteriori error, 159  
 predecessor selection, 290  
 predictive framework, 198  
 principal component projections, 329  
 principal components analysis, 24, 311  
 principal curve, 320, 324  
 principle of insufficient reason, 301  
 principle of relevant information (PRI), 300, 315, 316  
 priori error, 159  
 probability density function, 47  
 probability mass function, 11, 49  
 probability of classification error, 254  
 processing element, 182  
 projected power, 423  
 properties of EEC, 111  
 properties of Renyi's nonparametric entropy, 60  
 properties of the KL divergence, 17  
 proximity graph, 286  
 pseudometric, 128  
 pure rotation matrix, 249  
  
 QMI, 235, 246  
 QMIs, 89, 90, 93, 233, 485

- QMI<sub>ED</sub>, 89, 90, 93, 233, 235, 239, 242, 243, 359, 485  
 quadratic cross-entropy, 38  
 quadratic divergences, 84, 266  
 quadratic function, 114  
 quadratic information potential, 143  
 quadratic information potential estimator, 57  
 quadratic mutual information, 89, 306, 308  
 quadratic mutual information Euclidean distance (QMI<sub>ED</sub>), 86  
 quadratic Renyi's cross-entropy, 88  
 quadratic Renyi's entropy, 55, 58, 73, 124, 253  
 quadratic Renyi's entropy estimator, 56  
 quantum theory, 77  
 quasi-Newton methods, 188  
 radial basis function, 205, 220  
 radial conditional density function, 394  
 radial marginal, 394  
 random processes, 34  
 rate distortion theory, 9, 300, 313  
 receiver operating characteristic, 404  
 recursive estimator, 144  
 recursive information potential, 231  
 recursivity, 14  
 redundancy, 11, 308  
 regression, 31, 123, 209  
 regularizer, 112  
 reinforcement learning, 30  
 relative entropy, 17, 84, 317, 483  
 relative measure of the error, 192  
 relevant information, 313  
 Renyi's postulates, 400  
 Renyi's  $\alpha$  divergence, 81, 85  
 Renyi's  $\alpha$  entropy, 20  
 Renyi's  $\alpha$  mutual information, 83  
 Renyi's  $\alpha$ -divergence, 83  
 Renyi's  $\alpha$ -norm, 228  
 Renyi's entropy, 6, 49, 53, 54, 62, 219, 247, 266, 283, 337  
 Renyi's entropy of continuous variables, 55  
 Renyi's entropy property, 51  
 Renyi's family of entropy, 254  
 Renyi's gain of information, 83  
 Renyi's information measure, 49  
 Renyi's quadratic cross-entropy, 269  
 Renyi's quadratic entropy, 21, 54, 88, 106, 205, 248, 267, 317, 358, 361, 415  
 Renyi's quadratic error entropy, 103  
 Renyi's relative entropy, 87  
 Renyi's seven postulates, 401  
 representer theorem, 43  
 reproducing kernel, 34  
 reproducing kernel Hilbert spaces, 6, 351  
 reproducing property, 36, 355  
 resilient backpropagation, 186, 245  
 resubstitution estimates, 472  
 RKHS, 34, 35, 42, 43, 376  
 RKHS  $H_v$ , 420  
 RKHS and ITL, 38  
 RKHS definitions, 36  
 RLS, 167  
 robust, 126  
 robust statistics, 124  
 robustness, 210, 386  
 robustness in maximum likelihood, 126  
 sample estimator of cross-correntropy, 386  
 saturating limit of QMIs, 486  
 scale invariant cost function, 336  
 scale invariant entropy, 479  
 scale parameter, 58  
 scale-invariant, 339  
 scale-invariant cost function, 64  
 scaled conjugate gradient, 188  
 scaling property, 61  
 SDR, 347  
 self-organization, 299  
 self-organization principles, 315  
 self-organized information-theoretic principles, 300  
 self-organizing map, 315  
 self-organizing principle, 23, 30  
 semiparametric, 457  
 semiparametric density estimation, 461  
 sensitivities, 182  
 series expansion approximations, 462  
 Shannon information, 348  
 Shannon's entropy, 5, 48, 53, 54, 62, 248, 255, 330  
 Shannon's information gain, 17, 83

- Shannon's mutual information, 84, 311  
 shifting property of the Gaussian function, 96  
 SIG(1), 148  
 sigmoid, 182  
 signal to interference ratio, 340  
 signal-to-distortion ratio, 446  
 Silverman's rule, 58, 113, 115, 161, 209, 211, 227, 406, 468  
 similarity, 385  
 similarity matrix, 74, 267  
 similarity measure, 131  
 similarity metrics, 431  
 simplex, 52, 85  
 single-input-single-output, 169  
 singular value decomposition, 438  
 skewness, 13  
 source coding theorem, 9  
 sparseness, 309  
 spatial coherence, 308  
 spectral clustering, 267  
 spectrum of Renyi information, 50  
 speed up learning, 186  
 spike trains, 376  
 spline functions, 35  
 splitting data estimate, 473  
 statistical descriptors of spike trains, 376  
 statistical independence, 310  
 statistical inference, 43  
 statistical mechanics, 313  
 steepest descent, 27, 109, 141  
 stepsize, 27, 117  
 stochastic approximation, 27, 248  
 stochastic gradient algorithm, 138, 150  
 stochastic gradient approximation, 172  
 stochastic gradient update, 137  
 stochastic information gradient, 146  
 stochastic processes, 41, 415  
 stochastically sampling, 294  
 Stoller split, 223  
 subspace, 229  
 sum of marginal Renyi's entropies, 343  
 super-Gaussian, 175  
 supervised learning, 22, 30, 263  
 support vector machines, 35, 231, 273, 352  
 SVM, 242, 374  
 SVM algorithm, 371  
 SVM optimization criterion, 352  
 switching scheme, 155  
 symmetric kernel, 280  
 synthetic aperture radar, 448  
 system tracking, 214  
 Taylor expansion, 169  
 TDNN, 199, 213, 244  
 template matcher, 242  
 time constants, 119  
 time delay neural network, 182  
 time series, 41  
 time-varying linear system, 215  
 Toeplitz matrix, 419  
 Toeplitz nature, 170  
 trace, 120  
 track the weights, 174  
 Tsallis entropy, 54  
 turning point of curvature, 155  
 uncertainty, 11  
 uncorrelatedness in feature space, 392  
 unifying learning paradigm, 30  
 unifying view of learning, 31  
 unsupervised learning, 22, 30, 263  
 valley seeking clustering, 264  
 variable stepsize algorithms, 152  
 variance, 13, 72, 264, 458  
 variational parameter, 314  
 varying mixing matrices, 342  
 volume of a graph, 293  
 von Neumann's entropy, 55  
 wave function, 77  
 weak test for independence, 392  
 weight SNR, 173  
 weighted least squares, 126, 127  
 weighted mean vectors, 374  
 weighted Parzen estimation, 230  
 weighted Parzen window, 374  
 weighted Parzen window estimators, 375  
 weighted variance of the errors, 138  
 whitening matrix, 344  
 whitening-rotation, 346  
 Wiener-Hopf solution, 44  
 window length, 164  
 within-cluster entropy, 267  
 WSNR, 216  
 Z-score, 410



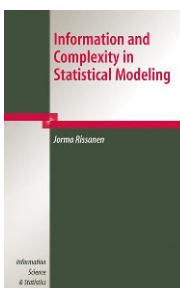
## Support Vector Machines

Ingo Steinwart

Andreas Christmann

**Content:** Preface.- Introduction.- Loss functions and their risks.- Surrogate loss functions.- Kernels and reproducing kernel Hilbert spaces.- Infinite samples versions of support vector machines.- Basic statistical analysis of SVMs.- Advanced statistical analysis of SVMs.- Support vector machines for classification.- Support vector machines for regression.- Robustness.- Computational aspects.- Data mining.- Appendix.- Notation and symbols.- Abbreviations.- Author index.- Subject index.- References.

2008. XVI, 602 p. 25 illus., 2 in color. Hardcover  
Information Science and Statistics  
ISBN: 978-0-387-77241-7



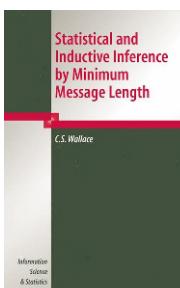
## Information and Complexity in Statistical Modeling

Jorma Rissanen

No statistical model is 'true' or 'false,' 'right' or 'wrong'; the models just have varying performance, which can be assessed. The main theme in this book is to teach modeling based on the principle that the objective is to extract the information from data that can be learned with suggested classes of probability models. The intuitive and fundamental concepts of complexity, learnable information, and noise are formalized, which provides a firm information theoretic foundation for statistical modeling.

**Content:** Introduction.- Shannon-Wiener information.- Coding with random processes.- Universal coding.- Kolmogorov complexity.- Stochastic complexity.- Structure function.- The MDL principle.- Applications.

2007. VIII, 144 p. Hardcover  
Information Science and Statistics  
ISBN: 978-0-387-36610-4



## Statistical and Inductive Inference by Minimum Message Length

C.S. Wallace

**Content:** Inductive Inference.- Information.- Strict Minimum Message Length (SMML).- Approximations to SMML.- MML: Quadratic Approximations to SMML.- MML Details in Some Interesting Cases.- Structural Models.- The Feathers on the Arrow of Time.- MML as a Descriptive Theory.- Related Work.- Bibliography. Index.

2005. XVI, 432 p. 22 illus. Hardcover  
Information Science and Statistics  
ISBN: 978-0-387-23795-4

### Easy Ways to Order ►

Call: Toll-Free 1-800-SPRINGER • E-mail: [orders-ny@springer.com](mailto:orders-ny@springer.com) • Write: Springer,  
Dept. S8113, PO Box 2485, Secaucus, NJ 07096-2485 • Visit: Your local scientific  
bookstore or urge your librarian to order.