Maximum-likelihood Kernel Density Estimation in High-dimensional Feature Spaces

Christiaan Maarten van der Walt

Maximum-Likelihood Kernel Density Estimation in High-dimensional Feature Spaces

by

Christiaan Maarten van der Walt

Thesis submitted for the degree

Philosophiae Doctor

in the

Department of Information Technology
Faculty of Economic Sciences

NORTH-WEST UNIVERSITY

Promoter: Professor Etienne Barnard

May 2014

Summary

Maximum-likelihood Kernel Density Estimation in High-dimensional Feature Spaces

by

Christiaan M. van der Walt

Promoter: Professor Etienne Barnard

Doctor of Philosophy

North-West University

Keywords: pattern recognition, non-parametric density estimation; kernel density estimation; kernel bandwidth estimation; maximum-likelihood; high-dimensional data; artificial data; probability density function.

With the advent of the internet and advances in computing power, the collection of very large high-dimensional datasets has become feasible – understanding and modelling high-dimensional data has thus become a crucial activity, especially in the field of pattern recognition. Since non-parametric density estimators are data-driven and do not require or impose a pre-defined probability density function on data, they are very powerful tools for probabilistic data modelling and analysis. Conventional non-parametric density estimation methods, however, originated from the field of statistics and were not originally intended to perform density estimation in high-dimensional features spaces – as is often encountered in real-world pattern recognition tasks. Therefore we address the fundamental problem of non-parametric density estimation in high-dimensional feature spaces in this study.

Recent advances in maximum-likelihood (ML) kernel density estimation have shown that kernel density estimators hold much promise for estimating nonparametric probability density functions in high-dimensional feature spaces. We therefore derive two new iterative kernel bandwidth estimators from the maximum-likelihood (ML) leave-one-out objective function and also introduce a new non-iterative kernel bandwidth estimator (based on the theoretical bounds of the ML bandwidths) for the purpose

of bandwidth initialisation. We name the iterative kernel bandwidth estimators the minimum leave-one-out entropy (MLE) and global MLE estimators, and name the non-iterative kernel bandwidth estimator the MLE rule-of-thumb estimator.

We compare the performance of the MLE rule-of-thumb estimator and conventional kernel density estimators on artificial data with data properties that are varied in a controlled fashion and on a number of representative real-world pattern recognition tasks, to gain a better understanding of the behaviour of these estimators in high-dimensional spaces and to determine whether these estimators are suitable for initialising the bandwidths of iterative ML bandwidth estimators in high dimensions. We find that there are several regularities in the relative performance of conventional kernel density estimators across different tasks and dimensionalities and that the Silverman rule-of-thumb bandwidth estimator performs reliably across most tasks and dimensionalities of the pattern recognition datasets considered, even in high-dimensional feature spaces. Based on this empirical evidence and the intuitive theoretical motivation that the Silverman estimator optimises the asymptotic mean integrated squared error (assuming a Gaussian reference distribution), we select this estimator to initialise the bandwidths of the iterative ML kernel bandwidth estimators compared in our simulation studies.

We then perform a comparative simulation study of the newly introduced iterative MLE estimators and other state-of-the-art iterative ML estimators on a number of artificial and real-world high-dimensional pattern recognition tasks. We illustrate with artificial data (guided by theoretical motivations) under what conditions certain estimators should be preferred and we empirically confirm on real-world data that no estimator performs optimally on all tasks and that the optimal estimator depends on the properties of the underlying density function being estimated. We also observe an interesting case of the bias-variance trade-off where ML estimators with fewer parameters than the MLE estimator perform exceptionally well on a wide variety of tasks; however, for the cases where these estimators do not perform well, the MLE estimator generally performs well.

The newly introduced MLE kernel bandwidth estimators prove to be a useful contribution to the field of pattern recognition, since they perform optimally on a number of real-world pattern recognition tasks investigated and provide researchers and practitioners with two alternative estimators to employ for the task of kernel density estimation.

Opsomming

Maksimumwaarskynlikheid-kerndigtheidskatting in Hoë-dimensionele Ruimtes

deur

Christiaan M. van der Walt

Promotor: Professor Etienne Barnard

Philosophiae Doctor

Noordwes-Universiteit

Kernwoorde: patroonherkenning, nie-parametriese digtheidskatting; kerndigtheidskatting; kernbandwydteskatting; maksimumwaarskynlikheid; hoëdimensionele data; kunsmatige data; waarskynlikeidsdigtheidsfunksie.

Die ontwikkeling van die internet en toename in rekenaarkrag het die versameling van baie groot hoë-dimensionele datastelle moontlik gemaak; gevolglik het die modellering en analise van hoë-dimensionele data hoogs noodsaaklik geword, veral in die veld van patroonherkenning. Nie-parametriese digtheidskatters is baie kragtige gereedskap vir die modellering en analise van data omdat hierdie skatters datagedrewe is en nie 'n parametriese vorm vir die digtheidsfunkie voorskryf nie. Konvensionele nie-parametriese dighteidskatters het hulle oorsprong in the veld van statistiek en is nie oorspronklik ontwikkel om digtheidsfunksies in hoë-dimensionale ruimtes te skat nie soos gereeld in patroonherkenning teëgekom word. Om hierdie rede ondersoek ons die fundamentele probleem van nie-parametriese digtheidskatting in hoë-dimensionele ruimtes in hierdie studie.

Onlangse deurbrake in maksimumwaarskynlikheid (MW) -kerndigtheidskatting het getoon dat kerndigtheidskatters baie belowend lyk om nie-parametriese waarskynlikheid-digtheidsfunksies te skat in hoë-dimensionele ruimtes. Om hierdie rede bewys ons twee nuwe iteratiewe kerndigtheidskatters vanuit die MW-raamwerk en ons stel 'n nuwe nie-iteratiewe kerndigtheidskatter (gebaseer op die teoretiese grense van MW-bandwydtes) voor. Ons noem die nuwe iteratiewe

kernbandwydteskatters die minimum-laat-een-uit-entropie (MLE) en die globale MLE-kernbandwydteskatter; ons noem die nie-iteratiewe kernbandwydteskatter die MLE-algemenereël-kernbandwydteskatter.

Ons vergelyk die akuraatheid van die MLE-algemenereëlbandwydteskatter en konvensionele kerndigtheidskatters op kunsmatige datastelle met data-einskappe wat sistematies gevarieer word; asook op regtewêrelddatastelle om die gedrag van konvensionele skatters in hoëdimensionele ruimtes beter te verstaan en om te bepaal of hierdie skatters geskik is om die bandwydtes van iteratiewe MW-digtheidskatters te inisialiseer in hoëdimensionele ruimtes. Ons vind dat daar verskeie reëlmatighede is in die relatiewe akuraathede van konvensionele kerndigtheidskatters tussen verskeie datastelle en dimensionaliteite en dat die Silverman-algemenereël-bandwydteskatter baie betroubaar presteer op die meeste van die patroonherkenningdatastelle wat ondersoek is, selfs in hoë-dimensionele ruimtes. Gegewe hierdie empiriese bewyse en die intuïtiewe teoretiese motivering dat the Silverman-skatter die asimptotiese gemiddelde geïntegreerde kwadratiese fout optimeer (gegewe 'n Gaussiese verwysingsverspreiding), maak ons gebruik van die Silverman-skatter om die bandwydtes van die iteratiewe MW-kernbandwydteskatters te inisialiseer in ons ondersoek.

Ons voer 'n vergelykende simulasiestudie uit tussen die nuut voorgestelde iteratiewe MLE-skatters en van die mees akkurate bestaande MW-skatters oor verkeie kunsmatige en regtewêreld-hoëdimensionelepatroonherkenning-datastelle. Ons illustreer met die kunsmatige data (gemotiveer deur teoretiese insigte) in watter toestande die verskeie skatters verkies moet word en ons bewys empiries op regtewêrelddata dat geen skatter optimaal presteer op alle datastelle nie en dat die optimaledigtheidskatter bepaal word deur die eienskappe van die onderliggende digtheidsfunksie wat geskat word.

Die nuut voorgestelde MLE-kernbandwydteskatters is 'n nuttige bydrae tot die veld van patroonherkernning, gegewe dat hierdie skatters optimaal presteer op versekeie regtewêreld-patroonherkenningdatastelle; die MLE-kernbandwydteskatters lewer dus twee alternatiewe metodes om kerndigtheidskatting te doen.

Table of Contents

CHAPTER ONE - Introduction 1

1.1 Introduction . . . 1
1.2 Problem Statement . . . 2
1.3 Research Objectives . . . 3
1.4 Outline of thesis . . . 4

CHAPTER TWO - Probability Density Estimation 5

2.1 Introduction . . . 6
2.2 Probability Density Estimation . . . 6
2.3 Non-Parametric Probability Density Estimation . . . 8
2.3.1 Histograms . . . 8
2.3.2 Naïve density estimation . . . 9
2.3.3 Kernel density estimation . . . 9
2.3.4 k-nearest-neighbour density estimation . . . 10
2.3.5 Variable kernel density estimation . . . 11
2.3.6 Projection pursuit density estimation . . . 12
2.3.7 Mixture model density estimation . . . 13
2.3.8 Bayesian networks . . . 15
2.4 Fitting criterion . . . 16
2.4.1 Goodness-of-fit for known distributions . . . 17
2.4.2 Goodness-of-fit for unknown distributions . . . 18
2.5 Kernel Density Estimation Considerations . . . 20
2.5.1 Selecting an appropriate kernel function for kernel density estimation 20
2.5.2 Conventional kernel bandwidth estimation approaches . . . 21
2.5.2.1 Rules of thumb . . . 22
2.5.2.2 Cross-validation methods . . . 23
2.5.2.3 Plug-in methods . . . 24

2.6 Statistical paired difference testing . . . 25
2.6.1 Paired-sample t-test . . . 26
2.6.2 Wilcoxon signed-rank test . . . 27
2.6.3 Wilcoxon rank-sum test . . . 27
2.6.4 Conclusion . . . 28

CHAPTER THREE - High-dimensional Data 29
3.1 Introduction . . . 29
3.2 High-Dimensional Pattern Recognition Data . . . 30
3.3 Properties of High-dimensional Data and the Curse of Dimensionality . . . 31
3.4 The Geometry of Feature Space . . . 32
3.5 Dimensionality Reduction . . . 34
3.6 Conclusion . . . 36

CHAPTER FOUR - Maximum-likelihood Kernel Bandwidth Estimation 37
4.1 Introduction . . . 38
4.2 Kernel Density Estimation . . . 38
4.3 Objective Functions for Optimisation of Bandwidth Selection Procedures . 39
4.4 Leave-one-out Likelihood Estimation . . . 40
4.5 Kernel Bandwidth Estimators . . . 40
4.5.1 MLE kernel bandwidth estimation . . . 41
4.5.1.1 Univariate Gaussian kernel . . . 42
4.5.1.2 Multivariate Gaussian kernel (full covariance matrix) . . . 42
4.5.1.3 Multivariate Gaussian Kernel (diagonal covariance matrix) 45
4.5.1.4 Global MLE estimator . . . 46
4.5.2 MLL kernel bandwidth estimation . . . 48
4.5.2.1 Univariate Gaussian kernel . . . 50
4.5.2.2 Multivariate Gaussian kernel (full covariance matrix) . . . 52
4.5.2.3 Multivariate Gaussian Kernel (diagonal covariance matrix) 53
4.5.3 ML-LOO kernel bandwidth estimation . . . 54
4.5.3.1 Spherical ML-LOO estimator . . . 54
4.5.3.2 Full covariance ML-LOO estimator . . . 56
4.5.4 Summary of ML KDEs computational complexities and parameters to be estimated . . . 57

4.6 Kernel Bandwidth Initialisation 58
4.7 Conclusion 59

CHAPTER FIVE - Comparison of Conventional Kernel Bandwidth Estimators 60
5.1 Introduction 61
5.2 Methods And Data 61
5.2.1 Data sampling procedure 61
5.2.2 Cross-validation data 63
5.2.3 Real-world data 64
5.2.4 Data pre-processing 66
5.2.5 Methods 68
5.2.6 Performance evaluation 68
5.3 Experimental design 69
5.3.1 Experiment 1 69
5.3.2 Experiment 2 71
5.3.3 Experiment 3 71
5.4 Results 71
5.4.1 Experiment 1 71
5.4.2 Experiment 2 81
5.4.3 Experiment 3 85
5.5 Conclusion 92

CHAPTER SIX - Comparison of Maximum-likelihood Kernel Bandwidth Estimators 95
6.1 Introduction 96
6.2 Methods And Data 97
6.2.1 Data Sampling Procedure 97
6.2.2 Cross-validation data 97
6.2.3 Real-world data 97
6.2.4 Data Pre-processing 97
6.2.5 Methods 97
6.2.6 Kernel bandwidth initialisation 97
6.2.7 Kernel bandwidth regularisation 99

6.2.8 Selecting the optimal number of training iterations 99
6.2.9 Performance evaluation 100
6.2.10 Statistical difference testing 100
6.3 Experimental Design 102
6.3.1 Experiment 1 102
6.3.2 Experiment 2 102
6.3.3 Experiment 3 103
6.4 Results 103
6.4.1 Experiment 1 103
6.4.2 Experiment 2 114
6.4.3 Experiment 3 118
6.5 Analysis of MLE AND MLE(g) Bandwidths 143
6.6 Conclusion 155

CHAPTER SEVEN - Conclusion 158
7.1 Introduction and Thesis Summary 158
7.2 Summary of Findings 160
7.3 Summary of Contributions 162
7.4 Suggestions for Further Research 163
7.5 Conclusion 165

REFERENCES 167

APPENDIX A - Conventional Estimator Results (Global PCA) 172
A.1 Introduction 172
A.2 CV Global PCA Results 173
A.3 RW Global PCA Results 175
A.4 Conclusion 180

APPENDIX B - Comparison of MLE and Silverman Initialisation 181
B.1 Introduction 181

B.2 Comparison of MLE rule-of-thumb and Silverman rule-of-thumb Initialisation 182
B.2.1 CV dataset results 182
B.2.2 RW dataset results 186
B.3 Conclusion 190

APPENDIX C - Maximum-likelihood Estimator Results (Global PCA) 192
C.1 Introduction 192
C.2 CV Global PCA Results 193
C.3 RW Global PCA Results 195
C.4 Convergence Examples of ML Estimators 207
C.5 Conclusion 212

List of Figures

5.1 *Test entropy results for Artificial Set 1 (D 1-10)* 73
5.2 *Test entropy results for Artificial Set 1 (D 10-1000)* 73
5.3 *Test entropy results for Artificial Set 2 (D 1-10)* 74
5.4 *Test entropy results for Artificial Set 2 (D 10-100)* 75
5.5 *Test entropy results for Artificial Set 3 (D 1-10)* 75
5.6 *Test entropy results for Artificial Set 3 (D 10-100)* 76
5.7 *Test entropy results for Artificial Set 4 (D 1-10)* 77
5.8 *Test entropy results for Artificial Set 4 (D 10-10)* 78
5.9 *Test entropy results for Artificial Set 5 (D 1-10)* 79
5.10 *Test entropy results for Artificial Set 5 (D 1-10)* 79
5.11 *Test entropy results for Artificial Set 6 (D 1-10)* 80
5.12 *Test entropy results for Artificial Set 6 (D 10-100)* 80
5.13 *Test entropy results for Artificial Set 7 (D 1-10)* 81
5.14 *Test entropy results for Artificial Set 7 (D 10-100)* 82
6.1 *Test entropy results for Artificial Set 1 (D 1-10)* 105
6.2 *Test entropy results for Artificial Set 1 (D 10-100)* 105
6.3 *Test entropy results for Artificial Set 2 (D 1-10)* 106
6.4 *Test entropy results for Artificial Set 2 (D 10-100)* 106
6.5 *Test entropy results for Artificial Set 3 (D 1-10)* 107
6.6 *Test entropy results for Artificial Set 3 (D 10-100)* 108
6.7 *Test entropy results for Artificial Set 4 (D 1-10)* 109
6.8 *Test entropy results for Artificial Set 4 (D 10-100)* 109
6.9 *Test entropy results for Artificial Set 5 (D 1-10)* 110
6.10 *Test entropy results for Artificial Set 5 (D 10-100)* 111
6.11 *Test entropy results for Artificial Set 6 (D 1-10)* 112
6.12 *Test entropy results for Artificial Set 6 (D 10-100)* 112
6.13 *Test entropy results for Artificial Set 7 (D 1-10)* 113
6.14 *Test entropy results for Artificial Set 7 (D 10-100)* 113
6.15 *Letter entropy results for training iterations(Class 15, CSPEC, PCA5)* . . 137

6.16 *Letter entropy results for training iterations(Class 15, CSPEC,PCA1)* . . . 138
6.17 *Segmentation entropy results for training iterations (Class 1, CSPEC, PCA5)*139
6.18 *Segmentation entropy results for training iterations (Class 1, CSPEC, PCA1)*139
6.19 *Landsat entropy results for training iterations (Class 3, CSPEC, PCA5)* . 140
6.20 *Landsat entropy results for training iterations (Class 3, CSPEC, PCA1)* . 140
6.21 *Landsat entropy results for training iterations (Class 4, CSPEC, PCA5)* . 141
6.22 *Landsat entropy results for training iterations (Class 4, CSPEC, PCA1)* . 142
6.23 *Landsat entropy results for training iterations (Class 13, CSPEC, PCA5)* . 142
6.24 *Landsat entropy results for training iterations (Class 13, CSPEC, PCA1)* . 143
6.25 *Letter MLE and MLE(g) bandwidths (class-specific PCA5)* 145
6.26 *Letter MLE and MLE(g) bandwidths (class-specific PCA1)* 146
6.27 *Segmentation MLE and MLE(g) bandwidths (class-specific PCA5)* 147
6.28 *Segmentation MLE and MLE(g) bandwidths (class-specific PCA1)* 147
6.29 *Landsat MLE and MLE(g) bandwidths (class-specific PCA5)* 148
6.30 *Landsat MLE and MLE(g) bandwidths (class-specific PCA1)* 148
6.31 *Optdigits MLE and MLE(g) bandwidths (class-specific PCA5)* 149
6.32 *Optdigits MLE and MLE(g) bandwidths (class-specific PCA1)* 150
6.33 *Isolet MLE and MLE(g) bandwidths (class-specific PCA5)* 153
6.34 *Isolet MLE and MLE(g) bandwidths (class-specific PCA1)* 154
B.1 *Balance Scale entropy results as likelihoods are removed (Class 1, class-specific PCA1)* 184
B.2 *Balance Scale entropy results as likelihoods are removed (Class 3, class-specific PCA1)* 185
C.1 *Letter entropy results for training iterations (Class 15, Global, PCA5)* . . . 207
C.2 *Letter entropy results for training iterations (Class 15, Global, PCA1)* . . . 208
C.3 *Segmentation entropy results for training iterations (Class 1, Global, PCA5)* 208
C.4 *Segmentation entropy results for training iterations (Class 1, Global, PCA1)* 209
C.5 *Landsat entropy results for training iterations (Class 3, Global, PCA5)* . . 209
C.6 *Landsat entropy results for training iterations (Class 3, Global, PCA1)* . . 210
C.7 *Optdigits entropy results for training iterations (Class 4, Global, PCA5)* . . 210
C.8 *Optdigits entropy results for training iterations (Class 4, Global, PCA1)* . . 211
C.9 *Isolet entropy results for training iterations (Class 13, Global, PCA5)* . . . 211
C.10 *Isolet entropy results for training iterations (Class 13, Global, PCA1)* . . . 212

List of Tables

4.1 Summary of computational complexities and parameters to be estimated for ML KDEs . . . 58
5.1 CV dataset summary . . . 65
5.2 RW dataset summary . . . 66
5.3 Summary of artificial dataset properties . . . 70
5.4 CV entropy results (class-specific PCA5) . . . 83
5.5 CV entropy results (class-specific PCA1) . . . 84
5.6 Letter test entropy results (class-specific PCA5) . . . 86
5.7 Letter test entropy results (class-specific PCA1) . . . 87
5.8 Segmentation test entropy results (class-specific PCA5) . . . 88
5.9 Segmentation test entropy results (class-specific PCA1) . . . 88
5.10 Landsat test entropy results (class-specific PCA5) . . . 89
5.11 Landsat test entropy results (class-specific PCA1) . . . 89
5.12 Optdigits test entropy results (class-specific PCA5) . . . 90
5.13 Optdigitstest entropy results (class-specific PCA1) . . . 90
5.14 Isolet test entropy results (class-specific PCA5) . . . 91
5.15 Isolet test entropy results (class-specific PCA1) . . . 92
6.1 CV entropy results (class-specific PCA5) . . . 114
6.2 CV entropy results (class-specific PCA1) . . . 115
6.3 Letter test entropy results (class-specific PCA5) . . . 119
6.4 Letter test entropy results (class-specific PCA1) . . . 120
6.5 Letter statistical difference test results (class-specific PCA5) . . . 122
6.6 Letter statistical difference test results (class-specific PCA1) . . . 123
6.7 Segmentation test entropy results (class-specific PCA5) . . . 124
6.8 Segmentation test entropy results (class-specific PCA1) . . . 124
6.9 Segmentation statistical difference test results (class-specific PCA5) . . . 125
6.10 Segmentation statistical difference test results (class-specific PCA1) . . . 125
6.11 Landsat test entropy results (class-specific PCA5) . . . 126
6.12 Landsat test entropy results (class-specific PCA1) . . . 126

6.13 Landsat statistical difference test results (class-specific PCA5) 127
6.14 Landsat statistical difference test results (class-specific PCA1) 127
6.15 Optdigits test entropy results (class-specific PCA5) 128
6.16 Optdigits test entropy results (class-specific PCA1) 129
6.17 Optdigits statistical difference test results (class-specific PCA5) 130
6.18 Optdigits statistical difference test results (class-specific PCA1) 130
6.19 Isolet test entropy results (class-specific PCA5) 132
6.20 Isolet test entropy results (class-specific PCA1) 133
6.21 Isolet statistical difference test results (class-specific PCA5) 134
6.22 Isolet statistical difference test results (class-specific PCA1) 135
6.23 RW dataset summary for selected classes 136
A.1 CV entropy results (global PCA5) 173
A.2 CV entropy results (global PCA1) 174
A.3 Letter test entropy results (global PCA5) 175
A.4 Letter test entropy results (global PCA1) 176
A.5 Segmentation test entropy results (global PCA5) 176
A.6 Segmentation test entropy results (global PCA1) 177
A.7 Landsat test entropy results (global PCA5) 177
A.8 Landsat test entropy results (global PCA1) 177
A.9 Optdigits test entropy results (global PCA5) 177
A.10 Optdigits test entropy results (global PCA1) 178
A.11 Isolet test entropy results (global PCA5) 178
A.12 Isolet test entropy results (global PCA1) 179
B.1 CV entropy results (MLE initialisation)(class-specific PCA1) 183
B.2 Letter test entropy results (MLE initialisation)(class-specific PCA1) 186
B.3 Segmentation test entropy results (MLE initialisation)(class-specific PCA1) 187
B.4 Landsat test entropy results (MLE initialisation)(class-specific PCA1) . . . 187
B.5 Optdigits test entropy results (MLE initialisation)(class-specific PCA1) . . 188
B.6 Isolet test entropy results (MLE initialisation)(class-specific PCA1) 189
C.1 CV entropy results (global PCA5) 193
C.2 CV entropy results (global PCA1) 194
C.3 Letter test entropy results (global PCA5) 195
C.4 Letter test entropy results (global PCA1) 196
C.5 Letter statistical difference test results (global PCA5) 197
C.6 Letter statistical difference test results (global PCA1) 198
C.7 Segmentation test entropy results (global PCA5) 198

C.8 Segmentation test entropy results (global PCA1) 199
C.9 Segmentation statistical difference test results (global PCA5) 199
C.10 Segmentation statistical difference test results (global PCA1) 199
C.11 Landsat test entropy results (global PCA5) 200
C.12 Landsat test entropy results (global PCA1) 200
C.13 Landsat statistical difference test results (global PCA5) 200
C.14 Landsat statistical difference test results (global PCA1) 200
C.15 Optdigits test entropy results (global PCA5) 201
C.16 Optdigits test entropy results (global PCA1) 201
C.17 Optdigits statistical difference test results (global PCA5) 202
C.18 Optdigits statistical difference test results (global PCA1) 202
C.19 Isolet test entropy results (global PCA5) 203
C.20 Isolet test entropy results (global PCA1) 204
C.21 Isolet statistical difference test results (global PCA5) 205
C.22 Isolet statistical difference test results (global PCA1) 206

List of Abbreviations

AMISE	asymptotic mean integrated squared error
AIC	Akaike information criterion
BIC	Bayesian information criterion
CV	cross-validation
EM	expectation maximisation
GMMs	Gaussian mixture models
iid	independent and identically distributed
ISE	integrated squared error
KDE	kernel density estimator
LLCV	local likelihood cross-validation
LOUT	leave-one-out
LSCV	least-squares cross-validation
MISE	mean integrated squared error
ML	maximum likelihood
MLE	minimum leave-one-out entropy
MLL	maximum leave-one-out likelihood
ML-LOO	maximum likelihood leave-one-out
MSE	mean squared error
MSP	maximum smoothing principle
MVN	multivariate normal
PDF	probability density function
PPDE	projection pursuit density estimation
RW	real-world
ULCV	uniform likelihood cross-validation

Mathematical Notation

X	a univariate random variable
x_i	the i^{th} sample of the univariate random variable X
$\mathbf{X}$	a multivariate random variable or a D-dimensional dataset
$\mathbf{x}_i$	the i^{th} sample of $\mathbf{X}$ represented by a D-dimensional vector
N	number of samples in dataset $\mathbf{X}$
D	number of dimensions in dataset $\mathbf{X}$
$K(x)$	a univariate kernel function evaluated at x
$K(\mathbf{x})$	a multivariate kernel function evaluated at $\mathbf{x}$
h	the bandwidth of a univariate kernel
$\mathbf{H}$	the bandwidth matrix of a multivariate kernel
$p(x)$	the univariate probability density function of X evaluated at x
$p(\mathbf{x})$	the multivariate probability density function of $\mathbf{X}$ evaluated at $\mathbf{x}$

Chapter One

Introduction

Contents

1.1 Introduction **1**

1.2 Problem Statement **2**

1.3 Research Objectives **3**

1.4 Outline of thesis **4**

1.1 INTRODUCTION

When data samples are observed in applications of statistics and pattern recognition a probability density function (PDF) of the observed data is typically estimated to mathematically describe the model that generated the observations. Important inferences regarding the properties of a set of observations can be made by investigating the estimated PDF of a dataset, such as: the distribution of the data, modality, scale variations, anisotropy and so on; this use of PDFs is usually found in statistics for the purpose of data analysis. PDFs are also used as tools for probabilistic reasoning, since they predict the likelihoods of observations occurring; this form of probabilistic reasoning is typically used in pattern recognition to make predictions for new observations based on previous examples. Probability density estimation is the process of estimating the PDF of observed data, and thus lies at the heart of data analysis and

model-based pattern recognition. Scientists have mainly focussed on defining and fitting predefined PDFs to data since the mathematical foundations for probability theory were laid though the study of games of chance by two scientists named Blaise Pascal and Pierre de Fermat in 1654 [1]. Since then many parametric distributions have been proposed as well as measures for goodness of fit to test how well observation fit an estimated distribution. This form of density estimation is known as parametric density estimation.

The strict distributional assumptions made in parametric density estimation restricted its use in practical applications, since data observed from real-world (RW) problems do not always follow a predefined parametric distribution. The first notable attempt to free discriminant analysis from strict distributional assumptions (where the distribution is not predefined but rather determined by the data) was made in 1951 by Fix and Hodges [2, 3] with the introduction of the naïve estimator. Kernel density estimators were formalised by Parzen in 1962 [4], which built on the work of Rosenblatt [5] and Whittle [6]. Since then, many approaches to non-parametric density estimation have been developed, most notably: k-nearest-neighbour density estimators [7], variable kernel density estimators [8], projection pursuit density estimators [9, 10], mixture model density estimators [11, 12, 13] and Bayesian networks [14, 15].

Since KDEs were formalised by Parzen, they have proven themselves to be useful tools for the task of nonparametric density estimation. However, it was not until the advent of the personal computer and the increase in computing power combined with the internet, that the collection of very large high-dimensional datasets became feasible. Scientists are no longer constrained by computing power and storage on computers, but now face the challenge of analysing and modelling massive datasets. The constraint thus moved from computing power and storage, to the ability of statistical techniques to function in very high dimensional spaces. The quest for developing statistical techniques for data analysis and modelling in high dimensional spaces has thus been initiated.

1.2 PROBLEM STATEMENT

Pattern recognition tasks cover a wide-range of applications and are often structured as high-dimensional feature sets. In this work, for example, we experiment with pattern recognition tasks in speech recognition, image segmentation, optical handwritten character recognition, letter recognition and satellite land cover classification, these tasks have datasets with dimensionalities of 617, 18, 62, 16 and 36 respectively. Since

conventional density estimation algorithms were developed in the field of statistics, they were not intended to perform density estimation in the high-dimensional features spaces often encountered in pattern recognition tasks [3, 16]. Scott [16, Sec 5.2] states, for example, that kernel density estimation of a full density function is only feasible up to six dimensions. We therefore define density estimation tasks with dimensionalities of 10 and higher as high-dimensional.

The behaviour of conventional density estimators on high-dimensional pattern recognition tasks is not fully understood, and has only recently been investigated in the context of such high-dimensional spaces for the purpose of pattern recognition[17, 18].

Recent advances in maximum-likelihood (ML) kernel density estimation have shown that KDEs hold much promise for estimating nonparametric density functions in high-dimensional spaces. Two notable contributions are the Maximum Leave-one-out Likelihood (MLL) KDE of Barnard [17] and the Maximum Likelihood Leave-One-Out (ML-LOO) estimator of Leiva-Murillo [18]. Both estimators were independently derived from the ML theoretical framework, the only differences being the simplifying assumptions made in their derivations to limit the complexity of the bandwidth optimisation problem. For both estimators, it was shown that non-parametric kernel density estimation can be performed in high-dimensional spaces with a reasonable degree of success.

The practical implications of the simplifying assumptions made in the derivations of the MLL and ML-LOO estimators on RW density estimation tasks are, however, not known since a derivation of a kernel bandwidth estimation technique without these assumptions does not exist.

1.3 RESEARCH OBJECTIVES

We address two fundamental problems in this work. First, we investigate the performance of conventional statistical density estimators on a number of representative pattern-recognition tasks, to gain a better understanding of the behaviour of these estimators in high-dimensional spaces. Second, we derive a kernel bandwidth estimation technique from the ML framework, similar to the approaches followed by Barnard [17] and Leiva-Murillo and Artés-Rodríguez [18]. The derivations of these approaches, however, contain simplifying assumptions to limit the complexity of the bandwidth optimisation problem.

In particular, the ML-LOO estimator by Leiva-Murillo and Artés-Rodríguez limits the number of free parameters in the bandwidth optimisation problem by estimating an identical full-covariance or spherical bandwidth matrix for all kernels. The MLL

estimator by Barnard assumes that the density function changes slowly throughout feature space, which limits the complexity of the objective function being optimised. Specifically, when the kernel bandwidth $\mathbf{H}_i$ for data point x_i is estimated, it is assumed that the kernel bandwidths of the remaining $N-1$ kernels are equal to $\mathbf{H}_i$. Therefore, the optimisation of the bandwidth $\mathbf{H}_i$ does not require the estimation of the bandwidths of the remaining *N-1* kernels.

The main *theoretical objective* of this study is to derive an ML kernel bandwidth estimator without the simplifying assumptions imposed by state-of-the-art ML kernel bandwidth estimators.

The main *empirical objectives* of this study are (1) to gain a better understanding of the behaviour of conventional kernel bandwidth estimators (specifically in the context of high-dimensional feature spaces), and (2) to investigate, on a number of representative pattern recognition tasks, whether the increased flexibility of a more general ML kernel bandwidth estimator will yield an improvement in performance over state-of-the-art ML kernel bandwidth estimators.

1.4 OUTLINE OF THESIS

The remainder of this thesis is structured as follows: Chapter 2 gives an overview of non-parametric density estimation and in Chapter 3 we discuss the non-intuitive implications of high-dimensional data. In Chapter 4 we describe the ML framework for kernel density estimation and we derive the state-of-the art ML kernel bandwidth estimation algorithms from first principles. We then give the proof of a novel kernel bandwidth estimation algorithm, derived from the same framework. In Chapter 5 we investigate the performances of conventional kernel bandwidth estimators on a number of artificial and RW datasets, and in Chapter 6 we compare the performances of ML kernel bandwidth estimation algorithms derived in Chapter 4. In Chapter 7 we make concluding remarks and suggest future work.

Chapter Two

Probability Density Estimation

Contents

2.1 Introduction 6

2.2 Probability Density Estimation 6

2.3 Non-Parametric Probability Density Estimation 8

2.3.1 Histograms 8

2.3.2 Naïve density estimation 9

2.3.3 Kernel density estimation 9

2.3.4 k-nearest-neighbour density estimation 10

2.3.5 Variable kernel density estimation 11

2.3.6 Projection pursuit density estimation 12

2.3.7 Mixture model density estimation 13

2.3.8 Bayesian networks 15

2.4 Fitting criterion 16

2.4.1 Goodness-of-fit for known distributions 17

2.4.2 Goodness-of-fit for unknown distributions 18

2.5 Kernel Density Estimation Considerations 20

2.5.1 Selecting an appropriate kernel function for kernel density estimation 20

2.5.2 Conventional kernel bandwidth estimation approaches 21
2.6 Statistical paired difference testing 25
2.6.1 Paired-sample t-test 26
2.6.2 Wilcoxon signed-rank test 27
2.6.3 Wilcoxon rank-sum test 27
2.6.4 Conclusion 28

2.1 INTRODUCTION

In Chapter 1 we briefly described the difference between parametric and non-parametric density estimation and pointed out that non-parametric density estimation is required for most RW applications to free density estimators from distributional assumptions. This is crucial since the underlying distribution of the data is typically not known in many RW applications and often does not follow any known parametric distribution.

In this chapter we define non-parametric density estimation more formally in Section 2.2, existing approaches to non-parametric density estimation are surveyed in Section 2.3, goodness-of-fit criteria are described in Section 2.4, detailed considerations for KDEs are discussed in Section 2.5, statistical paired difference tests that are relevant to our work are briefly described in Section 2.6 and we make concluding remarks in Section 2.7.

2.2 PROBABILITY DENSITY ESTIMATION

The probability distribution of a univariate random variable X with independent and identically distributed (*iid*) data samples $x_1, ..., x_N$ is defined as

$$P\left(a < X < b\right) = \int_a^b p(x)\, dx \qquad for\ all\ a < b \tag{2.1}$$

where $P\left(a < X < b\right)$ is the probability that a sample, x, from the distribution, X, will fall within the range $[a,\ b]$ and $p(x)$ is the PDF of X defined as

$$p(x) = \lim_{h \to 0} \frac{1}{2h} P\left(x - h < X < x + h\right) \tag{2.2}$$

where $P\left(x - h < X < x + h\right)$ can be estimated by the proportion of samples falling within the interval $[x - h,\ x + h]$.

The probability distribution of a D-dimensional multivariate random variable $\mathbf{X}$ with *iid* data samples $\mathbf{x}_1, ..., \mathbf{x}_N$ is defined as

$$P\left(a_1 < x_1 < b_1, ..., a_D < x_D < b_D\right) = \int_{a_D}^{b_D} \cdots \int_{a_1}^{a_D} p(x_1, ..., x_D)\, dx_1 ... dx_D \qquad for\ all\ a_i < b_i \tag{2.3}$$

where $\mathbf{P}\left(a < \mathbf{X} < b\right)$ is the probability that a sample, $\mathbf{x} = [\mathbf{x}_1, ..., \mathbf{x}_D]$, from the distribution, $\mathbf{X}$, will fall within the range $[\mathbf{a},\ \mathbf{b}]$ and $p(\mathbf{x})$ is the joint PDF of $\mathbf{X}$

$$p(x) = p(x_1, ..., x_D) = \lim_{h \to 0} \frac{1}{2h} P\left(x_1 - h < X_1 < x_1 + h, ..., x_D - h < X_D < x_D + h\right) \tag{2.4}$$

where $P\left(x_1 - h < X_1 < x_1 + h, ..., x_D - h < X_D < x_D + h\right)$ can be estimated by the proportion of samples that fall within the intervals specified for the D variables.

Probability density estimation is thus the process of estimating the unknown probability density function $p(\mathbf{x})$, from a set of observed data samples $\mathbf{X}$. If the parametric form of $p(\mathbf{x})$ is specified, the estimation process is known as parametric density estimation and if the parametric form of $p(\mathbf{x})$ is not specified, the estimation process is known as non-parametric density estimation.

Parametric density estimation is thus the process of estimating parameters required to fit a known parametric distribution to a set of data points. For example, if we assume that the observed dataset, X, was sampled from a univariate Gaussian distribution, the mean and standard deviation of the data must be estimated. These two quantities will provide a complete parametric description of the estimated univariate Gaussian distribution $p(X)$. Similarly, other parametric distributions can be fitted to an observed set of data points. One caveat, however, is that the parametric form of the distribution must be known prior to estimating the parameters of the distribution.

If a multivariate dataset $\mathbf{X}$ consists of only a few variables, the distribution of the data may be visualised in order to select an appropriate parametric distribution to fit to the data. When visualisation is not feasible, certain test are available to test how well the data fits a specified parametric distribution, e.g. the Kolomogorov-Smirnov test for multivariate normality [19]. This framework, however, limits the parametric form of the distribution fitted to the universe of known parametric forms. There are some cases when parametric density estimation is advantageous, e.g. when very few data samples are available and the parametric form of the distribution is known, then a good estimate of the parametric distribution can be obtained with very few data points, since only the parameters of the parametric distribution need to be estimated and not the shape of the distribution as well.

Non-parametric density estimation, on the other hand, does not require that the

parametric form of distribution be defined prior to density estimation. The density estimator is thus freed from distributional assumptions, with one major caveat: the distribution must be learned from the data. Thus, a sufficient number of representative data points are required in order to the estimate the true underlying distribution of the data accurately. The number of data points typically required to give a representative representation of the true distribution increases significantly with an increase in dimensionality of the feature space. We will give a detailed discussion of the implications of high-dimensional data in the next chapter. In the next section, we give a brief summary of the most widely used non-parametric density estimators in practice.

2.3 NON-PARAMETRIC PROBABILITY DENSITY ESTIMATION

The earliest approach to non-parametric density estimation is based on grouping data into bins, and counting the frequency of data points in each bin. This approach is known as the histogram approach and dates back to the 17th century [16]. The estimation of a histogram was formalised by Sturges in 1926 [20], and since then several refinements have been made to the histogram approach (such as frequency polygons [21] and average shifted histograms [22]) and a number of more advanced approaches to non-parametric density estimation have been developed. We will give a brief survey of these approaches in the remainder of this section. The textbook by Silverman [3] and tutorial by Scott [16] provides a more comprehensive survey of non-parametric density estimation techniques.

2.3.1 HISTOGRAMS

Histograms estimate the PDF of a D-dimensional dataset, $\mathbf{X}$, by partitioning the feature space into a pre-defined number of bins, K, and counting the number of data samples that fall within each bin. The density of a data point $\mathbf{x}_i$ falling into bin k is

$$\hat{p}(\mathbf{x}_i) = \frac{n_k}{\sum_{j=1}^{K} n_j vol_j} \tag{2.5}$$

where n_k is the number of data points grouped into bin k and vol_j is the volume of the j^{th} bin. If the D-dimensional feature space is divided into equally sized bins (with bin width h), vol_j will be h^D for all j. Note that the denominator is a normalisation factor that ensures that the density integrates to 1.

Rules to determine the appropriate bin width have been derived, such as the rules by Sturges [20] and Scott [16]. The unknown distribution (which we are trying to

estimate) is typically required in the derivations of these rules, and therefore these rules typically assume that the distribution is Gaussian, in order to derive practically useful rules.

The main drawback of histograms is that they are typically not feasible in high-dimensional spaces, since the number of bins increases exponentially with the number of dimensions, which creates numerous empty bins in high-dimensional spaces that will give probability density values of 0. For example, if each dimension is divided into 10 bins, then in 10 dimensions there will be 10^{10} bins, which implies that at least ten billion data points will be required to place a single data point in each bin.

2.3.2 NAÏVE DENSITY ESTIMATION

The naïve estimator was proposed by Fix and Hodges in 1951 [2] to free discriminant analysis from strict distributional assumptions. The naïve estimator places a rectangular region with width $2h$ and height $1/2Nh$ over each data point, where h is a bin width selected by the user, and N is the number of data points. The density of any data point x, is then estimated by summing the contributions $1/(2nh)$ of all the rectangular regions within which the data point falls. The density estimate is defined as

$$\hat{p}(x) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{h} w \left(\frac{x - x_i}{h} \right) \tag{2.6}$$

where

$$w(x) = \begin{cases} 1/2 & if\ |x| < 1 \\ 0 & otherwise \end{cases} \tag{2.7}$$

This approach is similar to the histogram approach where a bin centre is placed over each data point, and eliminates the need for the user to select appropriate bin positions. The naïve estimator, however, has three major drawbacks: (1) the estimated PDF is not continuous (since rectangular regions are summed), (2) if we want to obtain the density of an unseen test point, the data point may fall outside all of the rectangular regions fitted over the training points, which will give a probability density value of 0, and (3) the bin width, h, must be specified.

2.3.3 KERNEL DENSITY ESTIMATION

The KDE was proposed in 1958 by Ronsenblat [5] to overcome the non-continuous nature of the probability density estimate produced by the naïve estimator, by replacing the rectangular weight function in the naïve estimator with a kernel function. The kernel estimator thus centres a kernel function over each data point and sums

the contributions of the kernels to estimate the density of a data point $\mathbf{x}$. If the kernel function is non-negative and is a PDF itself, the estimated PDF will inherit the smoothness properties (continuity and differentiability) of the kernel function and will also be PDF [3, Sec. 2.4]

More formally, a KDE estimates the density function of a D-dimensional dataset $\mathbf{X}$, sampled *iid* from an unknown PDF, $p(\mathbf{x})$, with the sum

$$\hat{p}(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} K_{\mathbf{H}}\left(\mathbf{x}_i, \mathbf{x}\right) \tag{2.8}$$

where $K_{\mathbf{H}}(\cdot)$ is the kernel function with bandwidth matrix $\mathbf{H}$, $\mathbf{x}$ is the data point for which a density is estimated and $\mathbf{x}_i$ is the kernel centred at data point i in the training set.

If the kernel function satisfies

$$\int_{-\infty}^{\infty} K_{\mathbf{H}}\left(\mathbf{x}_i, \mathbf{x}\right) d\mathbf{x} = 1 \tag{2.9}$$

and

$$K_{\mathbf{H}}\left(\mathbf{x}_i, \mathbf{x}\right) \geq 0 \qquad for\ all\ \mathbf{x}_i\ and\ \mathbf{x} \tag{2.10}$$

the density estimate in Eq. 2.8 will be a PDF, and will inherit the smoothness properties of the kernel function.

Since the bandwidth matrix $\mathbf{H}$ is fixed across all kernels, kernel estimators are not able to model data accurately with variable scale. For example, if data points are dense in one region of the feature space, and sparse in another region, the estimator will over smooth in the dense region and under smooth in the sparse region. This typically leads to under smoothing of densities in the tails, since data points in the tails are typically very sparse [3, Sec. 2.4]. Variable kernel estimators attempt to address this shortcoming by adapting the bandwidth of an estimator throughout the feature space, and will be discussed later in this section.

2.3.4 K-NEAREST-NEIGHBOUR DENSITY ESTIMATION

The k-nearest-neighbour estimator adapts the density estimate throughout feature space based on local variations in scale, and was proposed by Loftsgaarden [7] in 1965. The estimated density of a data point $\mathbf{x}$ is adapted based on the volume of feature space required around point $\mathbf{x}$ to capture the k-nearest neighbour. More formally, the density is defined as

$$\hat{p}(\mathbf{x}) = \frac{k/N}{vol_k(\mathbf{x})} \tag{2.11}$$

where k is the pre-defined number of nearest-neighbours, N is the total number of data points in the D-dimensional training dataset $\mathbf{X}$ and $vol_k(\mathbf{x})$ is the volume of the D-dimensional sphere with radius equal to the Euclidean distance from $\mathbf{x}$ to its k^{th} nearest neighbour; this volume is defined as

$$vol_k(\mathbf{x}) = c_d[d_k(\mathbf{x})]^D \tag{2.12}$$

where $d_k(\mathbf{x})$ is the distance between $\mathbf{x}$ and its k^{th} nearest neighbour and c_d is the volume of the D-dimensional unit sphere.

The k-nearest neighbour density estimate is not a smooth curve because of discontinuities in the derivative of $d_k(\mathbf{x})$, and the estimate is not a PDF since it does not integrate to one [3, Sec. 2.5].

2.3.5 VARIABLE KERNEL DENSITY ESTIMATION

Another approach that adapts the density estimate based on local scale variations in feature space is the variable kernel method introduced by Breiman [8, 23] in 1977. The variable kernel method is similar to the kernel method, except that the bandwidths of kernels are unique (as opposed to having an identical kernel matrix for all kernels).

More formally, variable kernel estimators are defined as

$$\hat{p}(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} K_{h_{ik}}(\mathbf{x}_i, \mathbf{x}) \tag{2.13}$$

where $h_{ik} = hd_k(\mathbf{x})$ is the bandwidth matrix for kernel i, that is adapted based on the global smoothing parameter h and the distance to the k^{th} nearest neighbour $d_k(\mathbf{x})$.

The variable kernel estimator thus adapts the degree of smoothing based on local scale, inherits the smoothness properties (continuity and differentiability) of the selected kernel function (as opposed to the k-nearest-neighbour method), and does not suffer from zero likelihood scores if an unbounded kernel such as the Gaussian kernel is selected (as opposed to the naïve estimator). Similar to the standard kernel estimator, a smoothing parameter h must be provided by the user, the smoothing parameter is then scaled for each kernel with the distance to the k^{th} nearest neighbour $d_k(\mathbf{x})$. This variable kernel density estimator thus still requires the choice of h and k as design parameters.

Note that the kernel function K is radially symmetric, i.e. the same bandwidth is estimated per data point across all D dimensions. Consequently, the same scaling is used in all dimensions. This formulation thus generally works best if the data is whitened as a pre-processing step to account for the variation in scale between

dimensions. The variable kernel estimator this accounts for scale variations within features but not for scale variations between features. For example, if a certain part of the feature space $x = [\mathbf{x}_1, \mathbf{x}_2]$ has low density around $\mathbf{x}_1$ and high density around $\mathbf{x}_2$ (after the data have been whitened), then these variations in scale across dimensions will not be captured.

Numerous approaches to bandwidth estimation for kernel density estimation have been proposed since the introduction of the variable kernel density estimator. These approaches have been derived from various optimisation criteria, and make different assumptions in the derivation of the bandwidth estimation objective function. We will elaborate on these approaches in Section 2.5.

2.3.6 PROJECTION PURSUIT DENSITY ESTIMATION

Projection pursuit density estimators (PPDE) [9, 10] estimate the PDF of multivariate data as the product of univariate PDFs. Univariate PDFs (usually estimated with KDEs) are constructed from the "most interesting" orthogonal projections of the feature space, which are typically defined to be the most "non-normal" projections. An iterative approach is followed where the estimated multivariate PDF is updated with each iteration until a pre-defined stopping criterion is reached. The estimated density of a multivariate data point $\mathbf{x}$ is defined as

$$p_M(\mathbf{x}) = p_0(\mathbf{x}) \prod_{m=1}^{M} f_m \left(\theta_m \cdot \mathbf{x}\right) \tag{2.14}$$

where p_M is the density estimate after M iterations of the algorithm, p_0 is the initial density function which must be specified (typically a Gaussian distribution with sample mean and sample covariance), θ_m is a unit vector specifying the direction of interest and f_m is a univariate density function (typically estimated with a KDE).

PPDEs thus overcome the curse of dimensionality (described in Section 3.3) by restricting PDE to the univariate case and extending these estimates to higher dimensions. The main drawback of PPDEs is that the algorithm estimates a density function in the "most interesting" projected space. The PDF is thus not estimated in the original feature space, which is a requirement in certain applications.

2.3.7 MIXTURE MODEL DENSITY ESTIMATION

Mixture models estimate the PDF of observed data as a weighted sum of a specified parametric distribution

$$\hat{p}(\mathbf{x}_i) = \sum_{j=1}^{M} w_j \hat{p}_j(\mathbf{x}_i|\theta_j) \tag{2.15}$$

where M is the number of mixtures, $\hat{p}_j$ is the j^{th} mixture component with estimated parameters θ_j, and $\mathbf{x}_i$ is the data point for which the density is estimated.

Mixture models may thus be regarded as parametric estimators (on the one extreme) if the number of mixtures are specified *apriori*, and may be regarded as non-parametric estimators (on the other extreme) if the number of mixtures are automatically selected with a data-driven approach. When a Gaussian distribution is used as the parametric for of each mixture, the expectation maximisation (EM) algorithm [11, 12, 13] is typically used to estimate the mixture parameters θ_j, namely the mean and covariance matrix of each mixture. Kernel density estimators may also be regarded as a special case of mixture models, since each kernel centred on a data point can be regarded as a mixture, and each data point may be regarded as the centre of a mixture.

Mixture models, specifically Gaussian mixture models (GMMs), have been used with great success in practice, but suffer from two drawbacks. When the EM algorithm is used to estimate the mean and covariance of the various mixtures, there are specific cases where the covariance matrix is near singular i.e. may have at least one eigenvalue very close to zero. The second difficulty with mixture models is the selection of the appropriate number of mixture components. Various approaches, e.g. mixture splitting [24] and merging kernels into mixtures [25], have been proposed. However, these approaches still contain design parameters that must be set by the user to determine the degree of splitting or merging. The other alternative, often used in practice, is to specify the number of mixtures for the EM algorithm, and then iterate through a number of mixture components. Cross-validation (CV) can then be used to determine the optimal number of mixtures (based on the optimal performance, e.g. error rate for classification tasks or entropy for density estimation tasks). This approach is often used in classification tasks to select the number of mixtures of class-conditional density functions that optimises classification performance [26]. This approach is, however, computationally expensive (especially in light of the local minima that requires several random initialisations to be performed) and the user must still specify the range from which the optimal number of mixtures is selected.

Bayesian mixture models define the mixture weights and model parameters of a

mixture model as random variables and assign prior distributions over these random variables. Parametric Bayesian mixture models assign a parametric prior distribution to the mixture weights and parameters, while non-parametric Bayesian mixture models use stochastic processes (such as the Dirichlet Process (DP) which is an infinite sequence of probability distributions) to obtain non-parametric prior distributions for the mixture model parameters [27]. Note that the reference to parametric and non-parametric in the Bayesian context refers to the nature of the priors; the estimates of the probability distributions for both the parametric Bayes and non-parametric Bayes mixture models are non-parametric.

DPs are infinite sequences of discrete random variables or can be regarded as infinite sequence of discrete probability distributions. DPs provide a way to place non-parametric priors on mixture model parameters (thus freeing the user from selecting a parametric prior). DPs also have the additional advantage of providing an alternative to model selection for selecting the optimal number of mixtures for a mixture model by assuming an infinite number of mixtures. For finite datasets, however, the number of mixtures is finite and increases as the number of samples in the dataset increases.

A DP mixture model is constructed by repeatedly drawing a discrete distribution, G, from a DP, $DP(\alpha, G_0)$, with scaling parameter α (that controls the degree of variability of the drawn distributions, G, from the base distribution, G_0, where the base distribution is an arbitrary distribution that must be specified. The base distribution can be either continuous or discrete, but the distributions drawn from a DP will always be discrete. Parameters, η_n, are drawn from G and samples are drawn from distributions conditions on these parameters, $p(\cdot|\eta_n)$. If we repeatedly draw from DP, the parameters η_n will cluster into $\log(n)$clusters on average, owing to the fact that the draw from a DP will always be discrete, which will result in repeated values if enough draws are performed. On average $\log(n)$ distinct parameter values will exist (due to the clustering nature of DPs), and if we draw samples from distributions conditioned on these parameters, there will be on average $\log(n)$ mixtures.

The mixture model parameters can be obtained by inferring the posterior distributions of the mixture weights and parameters through either sampling methods or approximate variational inference methods. Approximate variational inference methods, such as mean-field variational inference [28], are preferred in high-dimensional feature spaces since they provide fast deterministic approximations, while Markov chain Monte Carlo approaches such as Gibbs sampling are more accurate but may become too computationally expensive in high-dimensional feature spaces to be useful in practice.

Since DP mixture models assume an infinite number of mixtures, they are also

referred to as infinite mixture models. However, for a finite dataset the number of mixtures will be finite owing to the clustering properties of the parameters drawn from discrete distributions drawn from a DP. Infinite mixture models were introduced by Rasmussen [29], who assumed multivariate Gaussian densities for the mixture components; these mixture models were called infinite GMMs. Rasmussen showed that DPs can provide an alternative to selecting the optimal number of mixtures by allowing an infinite number of mixtures. Rasmussen also introduced a sampling strategy (based on Gibbs sampling) to estimate the posterior mixture parameters of an infinite GMM.

In some applications where data is divided into groups, the sharing of mixtures between groups of data is required. DP mixture models have been extended to share mixture models between groups by defining a hierarchical DP, $DP(\alpha, H)$ from which a shared base distribution, G_0, is drawn [27]. For each of the J groups of data a base distribution, G_j, is drawn from the DP, $DP(\alpha_0, G_0)$, that is conditioned on the shared base distribution G_0; thus G_0 becomes the base distribution of the DP from which the base distribution, G_j, for each group is drawn, and α_0 controls the degree to which the base distributions for the groups varies from the shared base distribution. Similar to the standard DP, parameters are drawn from G_j for each of the J groups and samples are drawn from distributions conditioned on these parameters. The hierarchical DP thus serves as a prior distribution for the parameters drawn from G_j for each group and ensures that parameters (on which the data distributions are conditioned) are shared across groups, which results in shared mixtures across groups.

The sharing of mixtures across groups is very useful in applications such as topic modelling, where each document in a corpus can be regarded as a group, words in a document can be regarded as data points, a topic (which consists of a collection of words) can be regarded as a mixture and a document comprises a number of topics (or mixtures). Since the same topic (or mixture) can occur in numerous documents (or groups), the sharing of mixtures between groups is required. Hierarchical DPs thus allow for documents to be described as a mixture of topics, where the topics can be shared across documents in a corpus.

2.3.8 BAYESIAN NETWORKS

The joint PDF of a dataset may be expressed as the product of conditional density functions (based on the chain rule), in the following manner:

$$p(\mathbf{x}_1, ..., \mathbf{x}_D) = p(\mathbf{x}_D|\mathbf{x}_1, ..., \mathbf{x}_{D-1})p(\mathbf{x}_{D-1}|\mathbf{x}_1, ..., \mathbf{x}_{D-2})...p(\mathbf{x}_2|\mathbf{x}_1)p(\mathbf{x}_1) \tag{2.16}$$

where D is the number of variables in the feature space.

Bayesian networks attempt to simplify the conditional density functions in the factorisation of the joint PDF, by eliminating the modelling of some conditional dependencies between variables, e.g.

$$p(\mathbf{x}_D|\mathbf{x}_1, ..., \mathbf{x}_{D-1}) = p(\mathbf{x}_D|\mathbf{x}_1, \mathbf{x}_3) \tag{2.17}$$

Bayesian networks are typically represented graphically, where each node in a graph represents a variable, and the directional links between nodes represent conditional relationships between variables [14, 15]. Bayesian networks are thus visually appealing, since they give a graphical representation of a joint PDF and they explicitly indicate the conditional dependencies between variables.

Bayesian networks do, however, suffer from a significant drawback. The number of possible network configurations explodes exponentially as the number of variables increase. This limits their use in practice on high-dimensional data if the structure of the network is not known and must be learnt from data [30], since there will typically not be enough data to learn the optimal network topology in very high dimensions. If domain experts, however, specify the network structure (which includes the conditional dependencies between variables), this drawback may be circumvented; but this is typically a very time consuming process and may then be regarded as more of an expert system than a non-parametric density estimator of data.

2.4 FITTING CRITERION

There are broadly two approaches to measure how well a density estimator estimates the underlying PDF of a dataset, namely goodness-of-fit for known distributions (where the true PDF of the data is known or assumed) and goodness-of-fit measures for unknown distributions (where the true PDF of the data is not known or assumed).

Goodness-of-fit measures for known distributions are typically based on least-squares approaches that attempt to minimise the squared distance between the PDF estimated from a dataset and the true underlying PDF from which data were sampled. These measures include the mean-squared error (MSE), mean integrated squared error (MISE) and the asymptotic mean integrated squared error (AMISE). These measures work well for artificial data where data is sampled from a known distribution, but for RW data either a parametric form must be assumed for the distribution (called a reference distribution) or a pilot density must be constructed to get an initial PDF. To estimate the pilot density, however, comes down to the same initial problem of bandwidth estimation. Least-squares measures are very useful in deriving bandwidth

estimation algorithms for kernel density estimation, since these measures can be used as an objective function to minimise with respect to the bandwidths. We will discuss kernel bandwidth estimation approaches based on these measures in Section 2.5.

Goodness-of-fit measures for unknown distributions are typically based on ML approaches that attempt to maximise the product of the likelihood of each data point belonging to the estimated distribution (without knowing or assuming the true PDF of the data). The main drawback of the ML formulation is that there is a non-valid trivial solution at 0 for kernel bandwidth estimators, since the likelihood score of a data point falling directly onto a kernel centre with zero bandwidth will be infinite. We will show in Chapter 4 that this problem can be circumvented by making use of a leave-one-out (LOUT) ML estimation. Measures based on the likelihood score include the log-likelihood and entropy measures. Likelihood measures are also very useful for deriving kernel bandwidth estimators, since the LOUT ML measure can be used as an objective function to optimise with respect to the kernel bandwidths. We will discuss kernel bandwidth estimation approaches based on the ML measure in Section 2.5.

2.4.1 GOODNESS-OF-FIT FOR KNOWN DISTRIBUTIONS

The MSE measures the performance of an estimator by calculating the squared difference between the likelihood scores of a known distribution, $p(x)$, and an estimated distribution, $\hat{p}(x)$, at a specified location x. The mean of these differences is then calculated to estimate the MSE. More formally, the MSE is defined as

$$MSE_x = E\left[\hat{p}(x) - p(x)\right]^2 \tag{2.18}$$

where $E[\cdot]$ is the expected value operator, $p(x)$ is the known PDF of the data assumed to be square integral and $\hat{p}(x)$ is the estimated PDF of the data.

The MISE calculates how well the entire curve $\hat{p}(x)$ estimates $p(x)$ and is calculated as the expected value of the integrated squared distance between $p(x)$ and $\hat{p}(x)$ over all values of x. More formally, the MISE is defined as

$$MISE = E\int \left[\hat{p}(x) - p(x)\right]^2 dx \tag{2.19}$$

Note that since the integrant is non-negative the order of integration and the expectation operator can be reversed, and makes the MISE equivalent to the integrated MSE.

The MISE term can be expressed in an alternative form as the sum of an integrated squared bias and an integrated variance term [3, Sec. 3.3]. If we assume that $\hat{p}(x)$is

estimated with a kernel estimator, and that the kernel bandwidth h tends to 0 and the number of data points N tend to infinity, the bias and variance terms can be approximated with a Taylor series and results in

$$AMISE = \frac{1}{4}h^4 \left(\int t^2 K(t)dt \right) \int p''(x)^2 dx + \frac{1}{Nh} \int K(t)^2 dt \tag{2.20}$$

where h is the kernel bandwidth, $K(t)$ is a symmetric kernel that integrates to 1 and $p''(x)$ is the second derivative of the unknown density. The optimal bandwidth can thus be calculated by taking the derivative of the AMISE expression with respect to the bandwidth. Unfortunately the optimal bandwidth is also a function of the unknown density $p(x)$. The Silverman rule-of-thumb estimator derives a practically useful solution for the optimal bandwidth by substituting the unknown density, $p(x)$, with a standard normal distribution. The Maximal Smoothing Principle (MSP) bandwidth estimator is also derived from the AMISE. We will describe these estimators in more detail in Section 2.5.

2.4.2 GOODNESS-OF-FIT FOR UNKNOWN DISTRIBUTIONS

The task of estimating the goodness-of-fit of an estimated distribution, without knowing the true distribution from which the data was sampled is more challenging, since there is no reference distribution against which to compare the estimated distribution. The ML approach uses the product of the likelihood of each data point belonging to the estimated distribution (without knowing or assuming the true PDF of the data) as a measure for goodness of fit. More formally, the ML score of an estimated distribution $\hat{p}(\mathbf{x})$ for a D-dimensional dataset $\mathbf{X}$ with N *iid* samples, can be expressed as

$$L(\mathbf{X}) = \prod_{i=1}^{N} \hat{p}(\mathbf{x}_i) \tag{2.21}$$

We will show in Section 4.2 that the ML measure is analogous to the log-likelihood measure and that minimising entropy is equivalent to maximising the log-likelihood. We will also show in Section 4.3 how the likelihood measure can be restated in terms of the LOUT log-likelihood function to prevent the trivial solution, where h=0, that maximizes the likelihood function when $p(\mathbf{x})$ is estimated with the kernel estimator.

Alternative measures for goodness-of-fit for an unknown density function are the Bayesian Information Criterion (BIC), the Akaike Information Criterion (AIC) and Minimum Description Length (MDL). The BIC and AIC measures are also based on measuring the likelihood of observations, and penalise likelihood scores based on the

number of free parameters optimised in the estimator and the number of data points used in the estimate. The BIC measure is defined as

$$BIC = -2\ln\left(L\left(\mathbf{X}\right)\right) + k\ln\left(N\right) \tag{2.22}$$

where $L\left(\mathbf{X}\right)$ is the likelihood score as estimated in Eq. 2.21, k is the number of free parameters used in the estimation of $\hat{p}(\mathbf{x})$ and N is the number of data points used in the estimation.

Similarly, the AIC measure is defined as

$$AIC = 2k - 2\ln\left(L\left(\mathbf{X}\right)\right). \tag{2.23}$$

The AIC measure thus only penalises the model based on the number of free parameters and does not penalise the model for the number of data points used in the estimator.

The MDL criterion attempts to find a trade-off between the likelihood of data given a model and the model complexity. The MDL criterion selects the optimal model describing a dataset, $\mathbf{X}$, as the model that minimises the description length [31]

$$DL(\mathbf{X}, \theta) = DL(\mathbf{X}|\theta) + DL(\theta), \tag{2.24}$$

where $DL(\mathbf{X}|\theta)$ is the description length of the data, $\mathbf{X}$, given the model with parameters, θ, and $DL(\theta)$ is the description length of the model parameters.

If the model is assumed to be a PDF, $p(\mathbf{x})$, the description length of the data given $p(\mathbf{x})$, can be calculated with

$$DL(\mathbf{X}|\theta) = -\sum_{i=1}^{N} \log\left[p(x_i)\right], \tag{2.25}$$

since the optimal code assigns a code length of $\log\left[p(\mathbf{x})\right]$ according to Shannon's noiseless coding theorem [32]. Note that this description length is equivalent to sample entropy and the ML score and does not consider the model complexity.

The description length of the model, $DL(\theta)$, depends on the K parameters used to describe the model and on the quantisation of the K model parameters to a finite precision (if the parameters are continuous). Risannen [31] first introduced the MDL criterion and derived an ML approximation for the description length of the model that selects the optimal quantisation precision (assuming the model is a PDF $p(\mathbf{x}|\theta)$); this approximation is given by

$$DL(\theta) = K\log\left\|\theta\right\|_{M(\theta)} + \log\left[C(K)\right], \tag{2.26}$$

where K is the number of model parameters, θ is the parameter vector of length K, $M(\theta)$ is the $K \times K$ matrix of the second derivatives of $-\log[p(\mathbf{x}|\theta)]$, $\|\theta\|_{M(\theta)} = \sqrt{\theta' M(\theta)\theta}$ and $\mathrm{C}(K)$ is the volume of the K-dimensional unit hypersphere.

We note that a similar penalisation of model complexity is performed by making use of CV to measure performance. CV measures implicitly penalise models for overfitting, since the respective test points are left out of the estimation process and will have lower likelihood scores if the estimator overfits on the training data. For our experiments in Chapters 5 and 6 we will make use of entropy to measure the performance of kernel bandwidth estimators. This is equivalent to measuring the log-likelihood scores of the estimators (which will be shown in Section 4.2). We will make use of either CV or an independent test set to measure likelihood scores, which will implicitly penalise estimators for overfitting. CV and test set entropy thus gives a good measure of generalisation performance, since the data points used to obtain likelihood scores for the entropy calculation are not used in the the density estimate.

2.5 KERNEL DENSITY ESTIMATION CONSIDERATIONS

We have given a brief survey of the most notable approaches to non-parametric density estimation, during which we discussed the advantages and shortcomings of the various approaches. Specifically, we highlighted the theoretical and practical advantages of the kernel and variable kernel density estimators. We will discuss kernel bandwidth estimators in more detail in this section.

2.5.1 SELECTING AN APPROPRIATE KERNEL FUNCTION FOR KERNEL DENSITY ESTIMATION

The selection of the appropriate kernel function has both theoretical and practical considerations. In theory, the Epanechnikov kernel is optimal in an AMISE sense, since it has been derived by minimising the optimal AMISE with respect to the kernel function [33]. The efficiency of the Epanechnikov kernel have been compared to other symmetrical kernels (with respect to the MISE), and it has been shown in [3, Sec. 3.3] that the relative efficiencies of the Gaussian, Biweight, Triangular and Rectangular kernels do not differ significantly. Since the PDF produced by the kernel estimator will inherit the smoothness properties of the selected kernel, the choice of kernel should rather be based on mathematical properties (such as continuity and differentiability)

and practical considerations (such as not obtaining zero likelihood values for test data points that fall outside the region of training data points).

In practice, the Epanechnikov kernel has one significant shortcoming: it has a bounded tail. More specifically, an Epanechnikov kernel will only produce non-zero likelihood scores for values that fall within a distance of $\sqrt{5}$ from the kernel centre. In practice, this might lead to zero likelihood scores for test data points that do not fall within the radius covered by the Epanechnikov kernel of any training point, this will in turn lead to an infinite test entropy score. The Gaussian kernel, on the other hand is not bounded (thus zero likelihood scores will never occur for test points) and has an efficiency of 0.9512 relative to the efficiency of the Epanechnikov kernel (with respect to the MISE). The Gaussian kernel has another practical advantage over the Epanechnikov kernel, namely that the number of modes is monotonically non-increasing as the kernel bandwidth increases [16, Sec. 3.2].

Once an appropriate kernel function has been selected, selection of appropriate kernel bandwidths must be performed. We will discuss conventional approaches to kernel bandwidth estimation in more detail in the next section.

2.5.2 CONVENTIONAL KERNEL BANDWIDTH ESTIMATION APPROACHES

Kernel bandwidth estimation procedures are typically derived by optimising an objective function (that measures how well the data fits an estimated distribution) with respect to the kernel bandwidths. We have given a brief survey of the three most notable measures for goodness-of-fit that can be used as objective functions to optimise with respect to kernel bandwidths, namely the MSE, MISE and AMISE. In this section we will show how conventional kernel bandwidth estimators are derived from these measures.

Conventional approaches to kernel bandwidth estimation can be broadly categorised into rule-of-thumb, CV and plug-in approaches. *Rule-of-thumb* estimators are typically derived from the AMISE, and when certain assumptions are made regarding the unknown density function, analytical expressions can be derived for bandwidth estimation. The Silverman rule-of-thumb and MSP estimators assume a normal reference distribution, and fall into this category. *CV approaches* are typically based on either least-squares CV (LSCV) or ML CV. Least-squares approaches optimise the integrated squared error (ISE), and make use of a LOUT estimate in the approximation of the ISE between the true and estimated distribution. The LSCV estimator falls into this category. ML CV approaches are based on optimising the ML objective function. LOUT

CV is used to prevent the trivial solution to the optimisation of the ML bandwidth, where the bandwidth is 0 and the ML score is infinite. The likelihood cross-validation (LCV) and local LCV methods fall into this category. *Plug-in methods* attempt to find a closed-form solution for the optimisation of the AMISE with respect to the bandwidth. Similar to rule-of-thumb approaches, calculation of the optimal bandwidth requires the unknown distribution. Thus, instead of assuming a reference distribution for the unknown density, a pilot estimate of the unknown density is calculated. The pilot density, unfortunately, also requires the estimation of a bandwidth. The Hall Sheather Jones Marron (HSJM) estimator estimates the unknown functionals of f that are required in the calculation of the pilot bandwidth with a kernel estimator that makes use of normal rule-of-thumb bandwidths. This results in a closed form analytical expression for the optimal bandwidth. We will now give a more formal description of these estimators. To reflect conventional notation for a continuous distribution we will use f to denote the unknown distribution to be estimated, and $\hat{f}$ to denote the estimate of f.

2.5.2.1 RULES OF THUMB

The *Silverman rule-of-thumb* kernel bandwidth estimator optimises the AMISE with respect to the kernel bandwidth. The optimal bandwidth h_{Silv} can be derived from the AMISE by differentiating with respect to h and calculating the root of the derivative. This results in an expression for the optimal bandwidth that relies on the unknown distribution. It has been shown in [3, Sec. 3.4] that if the unknown distribution is assumed to be Gaussian and a Gaussian kernel is selected, then optimal bandwidth for the univariate case results in

$$h_{Silv} = 1.06\hat{\sigma}N^{-\frac{1}{5}} \tag{2.27}$$

where $\hat{\sigma}^2$ is the sample variance and N is the number of samples. To extend this rule to the multivariate case, the bandwidth can be estimated for each dimension independently.

The MSP kernel bandwidth estimator is derived in a similar fashion, but instead of assuming a Gaussian distribution for the unknown density function, a density f that minimises $R(f'')$ is assumed, where

$$R(f'') = \int (f(x)'')^2 \tag{2.28}$$

This will yield the upper bound of the optimal AMISE bandwidth [34] which gives the maximal smoothing compatible with the scale of the density estimate [35]. It has been shown [23] that the beta distribution minimises $R(f'')$if the variance is used as

scale parameter. If a Gaussian kernel is assumed, the bandwidth that maximizes the smoothing of the density estimate for a given scale, can be expressed as

$$h_{MSP} = 1.144\hat{\sigma}N^{-\frac{1}{5}} \tag{2.29}$$

We see in Eq. 2.27 and Eq. 2.29 that the estimated kernel bandwidths for Silverman and the MSP only differ by a factor of 1.08.

Jones, Marron and Sheather [36][34, Sec. 4a][35, Sec. 3.1] have studied the bandwidth estimate in Eq. 2.27 and have found that in the multivariate case, the Silverman estimate typically over smoothes the density estimate. We thus expect these bandwidth estimates to be slightly larger than the ideal bandwidth for high-dimensional data.

The MSP also over smoothes the density estimate since it has been designed to maximize the smoothing of a density estimate for a given scale. It has, however, been suggested by [23][34, Sec. 4a] that over smoothing does not force structure on the data, and forces us to rather find structure by further refining the bandwidths. These bandwidth estimates may thus be useful for the purpose of initialising iterative bandwidth estimation algorithms that will refine the initial bandwidth estimates.

2.5.2.2 CROSS-VALIDATION METHODS

LSCV procedures attempt to minimise the ISE. By using a method of moments estimate [37] the LSCV objective function is estimated as

$$LSCV(h) = \int \left(\hat{f}_h(y)\right)^2 dy - 2\sum_{i=1}^{N} \hat{f}_{-i,h}(x_i), \tag{2.30}$$

where $\hat{f}_{-i,h}(x)$ denotes the LOUT kernel density estimate with bandwidth h for data point y, when data point x_i is left out.

Eq. 2.30 shows that the objective function will require an iterative optimisation procedure and the selection of an initial bandwidth in to estimate the optimal LSCV bandwidth denoted by h_{LSCV}. We note that in practice the optimization problem becomes ill conditioned and practically infeasible if a unique bandwidth is estimated for each kernel, thus an identical bandwidth is typically estimate for all kernels.

For the multivariate case scaling of the features is performed by normalising features with their respective standard deviations. Bandwidths are estimated on the scaled data, and are re-scaled for the respective features after they have been determined. This approach results in a unique bandwidth for each feature.

The LSCV suffers from sample variability, thus the variability in bandwidths are high for different samples sampled from the same distribution [34, Sec. 4a]; the LSCV approach also suffers from local minima, which makes the performance of optimisation procedures that are susceptible to local minima, such as the Newton-Rhapson method, sub-optimal.

The *uniform LCV* estimator attempts to maximize the LOUT ML criteria (which we will define more formally in Chapter 4). Since the maximisation of the ML criteria has a trivial solution at $h = 0$, LOUT CV is employed to prevent this degenerate solution. The LOUT ML objective function is defined as

$$LCV(h) = \frac{1}{N} \sum_{i=1}^{N} \log \left(\hat{f}_{-i,h}(x_i) \right). \tag{2.31}$$

If a single bandwidth is estimated to optimise Eq. 2.31 the procedure is known as uniform LCV. The optimal uniform LCV bandwidth, denoted by h_{CV}, is thus the bandwidth that minimises the objective function in Eq. 2.31. Similar to the LSCV, a single bandwidth is employed everywhere, and the estimation of unique bandwidths for each kernel tend to make the optimisation problem ill-conditioned and practically infeasible. This approach also requires the estimation of an initial bandwidth to initialise the optimisation procedure, similar to the LSCV estimator.

The *local LCV* estimator makes the estimation of a unique bandwidth for each kernel practically feasible by circumventing the optimisation of numerous bandwidths simultaneously. The uniform LCV bandwidth for each kernel is adapted locally by scaling each bandwidth with the distance to its nearest-neighbour.

2.5.2.3 PLUG-IN METHODS

Plugin methods attempt to find a closed-form solution for the optimisation of the AMISE with respect to the bandwidth. Similar to rule-of-thumb approaches, calculation of the optimal bandwidth requires the unknown distribution f. Thus, instead of assuming a reference distribution for the unknown density, a pilot density is estimated to obtain an estimate for $R(f'')$ given in Eq. 2.28. If the pilot density is estimated with a kernel estimator, a kernel bandwidth is required, and for the HSJM plug-in rule it has been shown [38] that the pilot bandwidth, g, can be written as a function of the bandwidth, h, being estimated as

$$g(h) = C(K) \left[\frac{R(f'')}{R(f''')} \right]^{\frac{1}{7}} h^{\frac{5}{7}}. \tag{2.32}$$

where the constant $C(K)$ is determined by the kernel type, $K(t)$, and is given by

$$C(K) = \left[\int t^2 K(t)dt\right]^{\frac{2}{5}} \left(\int K(t)^2 dt\right)^{\frac{4}{5}}. \tag{2.33}$$

If the unknown functionals of f in Eq. 2.32 are solved with a kernel estimator that makes use of normal rule-of-thumb kernel bandwidths, the following expression is obtained

$$h = \left[\frac{R(K)}{\left(\int x^2 K(x)dx\right)^2 R(\hat{f}''_{g(h)})}\right]^{\frac{1}{5}} N^{-\frac{1}{5}}, \tag{2.34}$$

where $R(K)$ is given in Eq. 2.28, $K(\cdot)$ is the kernel function and $\hat{f}''_{g(h)}$ is the pilot kernel density estimate with bandwidth $g(h)$ given in Eq. 2.32.

Eq. 2.34 shows that the only variable to be solved is the bandwidth h and that h occurs on both sides of the equation owing to the dependence of the pilot bandwidth on h. To solve for the optimal value of h, denoted as h_{HSJM}, the objective function in Eq. 2.34 must be initialised with a value for h; the right-hand side of the equation can then be calculated which will yield an updated value for h. The updated value for h can then be substituted into the right-hand side of Eq. 2.34 again, and the process can be repeated iteratively until the bandwidth, h, converges to the optimal bandwidth h_{HSJM}.

2.6 STATISTICAL PAIRED DIFFERENCE TESTING

We will consider paired difference tests in this study to determine the statistical significance of the differences between estimator results. Paired difference tests are used to determine whether the means or medians of two sets of measurements are different, given a certain significance level. First, the difference between the paired measurements is calculated as

$$Z = X - Y \tag{2.35}$$

where Z is the paired difference of measurements X and Y.

The objective of a paired difference test is then to determine whether Z is sufficiently close to 0, given the confidence level. The selection of the appropriate difference test is dependent on the assumptions made regarding Z. If the paired differences, Z, are assumed to be normally distributed around the zero mean, then the paired Student's t-test may be used. If Z is not normally distributed around the zero mean, but is symmetrically distributed around the zero median, the Wilcoxon signed-rank test may be used. Typically, the Kolomogorov-Smirnov test statistic for MVN can be used to

test whether Z is normally distributed and the g statistic for skewness [39, 40] can be used to test whether Z is symmetric around the median. If Z is not symmetrically distributed around the 0 median, the Wilcoxon rank-sum test can be employed.

We will give a brief overview of the Student's t-test, the Wilcoxon signed-rank test and the Wilcoxon rank-sum test in the remainder of this section.

2.6.1 PAIRED-SAMPLE T-TEST

The paired-sample t-test (sometimes called the dependent t-test for paired samples or the paired student's t-test) can be performed by calculating

$$t = \frac{\mu_Z - \mu_0}{\sigma_Z / \sqrt{N}} \tag{2.36}$$

where μ_Z is the mean of the paired differences (Z), μ_0 is the hypothesized population mean (in this case zero) and N is the number of sample pairs in the test statistic. The degree of freedom used for the test is $N - 1$.

The null hypothesis that the sample Z comes from a normal distribution with mean equal to zero, with a 1% significance level ($\alpha = 0.01$) can be tested by comparing the t-statistic calculated in Eq. 2.36 with the critical values of a Student's t-distribution computed using the cumulative distribution function at the α significance level, with $N - 1$ degrees of freedom. Otherwise, the critical values of the Student's t-distribution can be read from a reference table [41]. Note that for a two-sided test the column corresponding to $1 - \alpha/2$ must be used, thus 0.995 for $\alpha = 0.01$. If the test statistic is greater than the critical value of $t_{1-\alpha/2,\nu}$ in the table, the null hypothesis is rejected for the specified significance level (the means are thus regarded as significantly different), else the null hypothesis is accepted (in which case we will regard the means as not significantly different).

Alternatively, the probability of the observation given the null hypothesis is true (p-value) can be calculated from the test statistic calculated in Eq. 2.36. If p is smaller than the probability threshold we reject the null hypothesis and regard the means of X and Y as significantly different. Otherwise, if p is equal or larger than the probability threshold, we accept the null hypothesis and regard the means of X and Y as not significantly different. Note that the probability threshold for the p-value is not necessarily the same as α, but both are regarded as significance levels and are typically set to 0.01. We prefer to make use of p values in our experiments, since they give a meaningful accept or reject hypothesis result (similar to the critical values approach), and the p value itself is meaningful as a probability that the means of X and Y are not significantly different for the confidence level provided.

Since the hypothesis that is tested with the paired-sample t-test, is whether Z is normally distributed and has zero mean, the hypothesis can be rejected if the distribution is not normal (even though the mean might be sufficiently close to zero). Therefore, a test for MVN must be performed to test whether Z is normally distributed, to ensure that the t-test result is valid. The Kolmogorov-Smirnov test for MVN is a reliable test statistic for multivariate normality [19].

2.6.2 WILCOXON SIGNED-RANK TEST

The Wilcoxon signed-rank-test follows a similar procedure to the Student's t-test. A test statistic for the paired-differences is calculated and then either compared to a critical value from a reference table, or alternatively a p-value can be calculated and compared to the probability threshold provided. The null hypothesis for the Wilcoxon signed-rank test is that the paired-differences have zero median. To test this hypothesis, the Wilcoxon test statistic is calculated as follows: first all zero values of the paired-difference test statistic, Z, are removed. The remaining values of Z are then ranked according to their absolute values, where a rank of 1 is assigned to the smallest absolute value and a rank of M is assigned to the largest absolute value, where M is the number of non-zero values in Z. Whenever two values in Z have the same value, they are assigned an average rank value. For example, if two values identical and are ranked as the second smallest value, the average rank assigned to both values will be 2.5. The ranks of all positive Z values are then summed to give the test statistic W^+. If there is a significant difference between the pairs X and Y, W^+ is expected to be far from its expected value, and is expected to converge to a normal distribution as the number of samples, M, increase. Therefore, a z-statistic of W^+ is typically calculated (when $M \geq 10$) and compared to a critical z-score value to accept or reject the null hypothesis at the specified significance level. If $M < 10$, W^+ is compared to a critical value in a reference, to determine if the null hypothesis should be accepted or rejected.

2.6.3 WILCOXON RANK-SUM TEST

The Wilcoxon rank-sum groups the values in X and Y together and ranks them from smallest to largest. The smallest value receives a rank of 1 and the largest value a rank of N (where N is the total number of samples in X and Y). The ranks of all values in X are then summed to give the test statistic W. If there is a significant difference between the pairs X and Y, W is expected to be far from its mean. Similar to the rank-sum test, a z-statistic can be calculated and compared to a critical value to accept

or reject the null hypothesis at the specified significance level.

2.6.4 CONCLUSION

In this chapter we provided an overview of the most notable approaches to non-parametric density estimation and discussed criteria that are typically used to evaluate the performance of density estimators. We then gave a more detailed discussion on conventional kernel density estimation and briefly described statistical paired difference tests that are relevant to our work.

Chapter Three

High-dimensional Data

Contents

3.1 Introduction 29

3.2 High-Dimensional Pattern Recognition Data 30

3.3 Properties of High-dimensional Data and the Curse of Dimensionality 31

3.4 The Geometry of Feature Space 32

3.5 Dimensionality Reduction 34

3.6 Conclusion 36

3.1 INTRODUCTION

In Chapter 1 we stated that pattern recognition tasks are often structured as high-dimensional feature sets, and that conventional density estimation algorithms (typically developed in the field of statistics), were not intended to perform density estimation in high-dimensional feature spaces often encountered in pattern recognition tasks [3, 16]. Scott [16, Sec 5.2], for example, states that density estimation beyond six dimensions with conventional approaches is often regarded as practically infeasible. We therefore defined high-dimensional data as data with dimensionalities of 10 and higher.

In the same light we mentioned that recent advances in pattern recognition have shown that density estimation is indeed possible on high-dimensional pattern recognition tasks [18]. We therefore devote this chapter to describing and illustrating the counter-intuitive nature of high-dimensional feature spaces.

In this chapter we give examples of high-dimensional pattern recognition tasks in Section 3.2, illustrate the counter-intuitive properties of high-dimensional data in Section 3.3, describe the geometry of feature spaces in Section 3.4, discuss dimensionality reduction approaches in Section 3.5 and make concluding remarks in Section 3.6.

3.2 HIGH-DIMENSIONAL PATTERN RECOGNITION DATA

Pattern recognition problems are typically structured as D-dimensional datasets, where each dimension represents a feature based on measurements that describe the problem of interest, and each sample is represented as a D-dimensional feature vector. From a geometric perspective, each D-dimensional feature vector may be regarded as a data point in a D-dimensional feature space; in this context each feature thus represents an axis of the feature space.

Pattern recognition covers a wide range of applications, and problems in pattern recognition are often structured as high-dimensional datasets. For example, in automatic speech recognition 39-dimensional feature vectors are constructed from regular time intervals of speech signals and these features are then used to describe phonemes in spoken language that are in turn used to recognise speech. In fingerprint recognition, Gabor filters with various orientations are typically used to filter a centralised region of a fingerprint and variances of sectors of these fingerprints are then used as features to describe a fingerprint. The dimensionality of features depends on the number of filter orientations and segments, and often comprises more than 100 dimensions. In this work, for example, we experiment with pattern recognition tasks in speech recognition, image segmentation, optical handwritten character recognition, letter recognition and satellite land cover classification. These datasets have dimensionalities of 617, 18, 62, 16 and 36 respectively.

There are numerous more such examples, as a quick survey of the UCI Machine Learning Repository [42] will show. The UCI Machine Learning Repository is one of the most comprehensive databases for pattern recognition data sets on the internet, and contains a collection of over 200 datasets contributed by scientists from various fields of research. A survey of this database indicates that approximately three in

four datasets have a dimensionality of 10 and more. This illustrates why data analysis techniques, such as density estimators, must be able to function in high-dimensional spaces to be practically useful for pattern recognition tasks.

3.3 PROPERTIES OF HIGH-DIMENSIONAL DATA AND THE CURSE OF DIMENSIONALITY

It has been shown in [3, Sec. 4.5] that intuitions that hold for one, two and three dimensions often do not hold for higher dimensions. We illustrate by several examples how lower-dimensional intuitions are often misleading for higher-dimensional data.

A first example illustrates how the lower-dimensional intuition that most of the mass of a standard normal distribution is centred on the mean is misleading. For a standard normal distribution, nearly 90% of the mass of the distribution falls within distance 1.6 from the centre in the one-dimensional case. For the 10-dimensional MVN distribution, however, less than 1% of the mass falls within distance 1.6 from the centre. This implies that the mass of an MVN density moves to the tails as dimensionality increases. This point is further illustrated by the fact that more than half of the samples of a 10-dimensional MVN distribution lie at density values less than a hundredth of the maximum density value. Thus, as dimensionality increases, more samples lie at the tails of the distribution (on average) and fewer samples occur near the centre of the distribution, thus resulting in large regions in the feature space that are relatively empty. This phenomenon is known as the *empty space phenomenon* [43, 44].

This phenomenon can also be illustrated geometrically by considering the ratio between the volume of a hyper-cube and a hyper-sphere with the same centre, where the diameter of the hyper-sphere is equal to the width of the hypercube. As dimensionality increases, the ratio between the volume of the hyper-sphere and hyper-cube reduces drastically. This again implies that as dimensionality increases, the volume of the feature space moves towards the outer edges.

The practical implication of the empty space phenomenon is that modelling the tail accurately becomes increasingly important as dimensionality increases, and is much more important than lower-dimensional intuitions make us believe.

A second example illustrates how lower-dimensional intuitions are misleading regarding the number of data points that are required to estimate a density function accurately. Silverman [3, Sec. 4.5] constructed a table for the standard MVN distribution containing the minimum number of samples sizes required to achieve a relative MSE < 0.1, for dimensionalities ranging from 1-10. More formally, the relative MSE

for a density $p(\mathbf{x})$ at data point 0 is defined as

$$\mathbf{E}\left\{\hat{p}(\mathbf{x}) - p(\mathbf{0})\right\}^2 / p(\mathbf{0})^2 \tag{3.1}$$

where $p(\mathbf{x})$ is the standard MVN density at location $\mathbf{x}$, and $\hat{p}(\mathbf{x})$ is the density at point $\mathbf{0}$ estimated with a Gaussian kernel for which the bandwidth has been selected to minimise the MSE at this point.

The table indicates that in one dimension, only four samples are required; in two dimensions 19 samples, in three dimensions 67 samples, in five dimensions 768 samples and in 10 dimensions 842 000 samples are required to achieve a relative MSE of less than 0.1. As is seen from this sequence, the number of required data points increases non-linearly as the dimensionality increases, and the non-linearity of this increase might not be so clear for dimensionalities up to five.

We use another example to illustrate how the volume of feature space increases exponentially as dimensionality increases. This leads to significant data sparsity, and is typically referred to as the *curse of dimensionality*. Consider, for example, how the numbers of bins of a histogram increase as the dimensionality increases. If each dimension contains 10 discrete intervals, we will require 10^D data points to place only a single data point in each bin. Thus, for three dimensions we will need a thousand data points and for 10 dimensions we will require ten billion data points.

These theoretical and geometrical examples all illustrate that our lower dimensional intuitions are often misleading in higher dimensions. These lessons imply that we should be mindful of these counter-intuitive examples when we develop and evaluate algorithms for pattern recognition based on lower dimensional intuitions. We will, however, explain in the next section why the intrinsic dimensionalities of datasets in practice are, fortunately, often of lower dimensionality than the feature spaces in which they are structured.

3.4 THE GEOMETRY OF FEATURE SPACE

In practice, pattern recognition algorithms are often successful in high-dimensional spaces, even in the absence of such a large number of samples that are, for example, suggested when finding the minimum required number of samples for a certain relative MSE. It is therefore reasonable to assume that data might be concentrated in localised regions with lower intrinsic dimensionalities than the feature space in which they are embedded.

A practical example to illustrate this is given by Bruske [45]. If we consider an example of a body suit with N sensors capturing motion in three dimensions, a feature space of dimensionality $3N$ is constructed if each measurement is regarded as a feature. The exact position of a body can actually be specified by k angles between the joints of the body. The intrinsic dimensionality of the problem (k) is thus significantly smaller than the dimensionality of the feature space ($3N$). Similarly, it has been illustrated by Lee, Pederson and Mumford [46] that high-dimensional natural images can (to a good approximation) be reduced to points on a 7-sphere. These properties of data imply that data points are not uniformly distributed throughout feature space and that data points are concentrated in small parts of the feature space that contribute to most data density.

These examples suggest that high-dimensional data can be described by manifolds with intrinsic dimensionalities k much lower than the dimensionality of the feature space (where we define intrinsic dimensionality as the minimum number of free parameters required to describe the data). We have proposed in [47] that the geometry of feature space can be defined with the following components: (1) the positions of manifolds in feature space, (2) the geometrical shapes of manifolds, and (3) the variation of data points from manifolds.

We note that KDEs capture these components implicitly, since (1) the positions of manifolds are modelled by kernel locations, (2) the geometrical shapes of manifolds are modelled by the combination of kernel locations and kernel bandwidths, and (3) the variation of data points from manifolds are modelled by kernel bandwidths that control the degree of smoothing of the density estimate, and consequently the data variation. Kernel estimators are thus capable of modelling the geometry of feature space (if density estimation is performed in the original feature space) and are capable of modelling the underlying manifold of the data (if density estimation is performed on a lower-dimensional feature space where features have been compressed). Kernel density estimates can also be used to visualise data by either visualising the density of projected features or visualising two-dimensional sections of a D-dimensional feature space; we refer the reader to [16, Sec. 3.4] and [48] for illustrations of these data visualisation techniques.

We have provided some examples in this section that indicate that RW pattern recognition tasks often have much lower intrinsic dimensionalities than the feature spaces in which they are structured. We will discuss dimensionality reduction techniques in the next section, since they are concerned with compressing data to lower dimensions while retaining as much information of the original feature space as possible.

It is also worth noting that dimensionality reduction is often structured as a manifold learning problem, corresponding to our intuition that RW data can be described by manifolds of lower dimensionality.

3.5 DIMENSIONALITY REDUCTION

Dimensionality reduction techniques attempt to project data in a feature space, $\mathbf{X}$, with dimensionality, D, to a lower dimensional feature space, $\mathbf{Y}$, with dimensionality k, where $k << D$. Geometrically, data points in dataset $\mathbf{X}$ are typically presumed to lie on or near a manifold with dimensionality k, that is embedded in a D-dimensional space [49]. Dimensionality reduction techniques thus attempt to transform the data from $\mathbf{X}$ to $\mathbf{Y}$, while retaining the geometry of the underlying k-dimensional manifold.

Dimensionality reduction approaches can be broadly classified into linear and non-linear approaches. Linear dimensionality reduction techniques perform a linear transformation of $\mathbf{X}$ to $\mathbf{Y}$, while non-linear techniques perform a non-linear transformation of $\mathbf{X}$ to $\mathbf{Y}$.

A comparative study was performed by van der Maaten [49], where linear principal component analysis (PCA) was compared to 12 state-of-the art non-linear dimensionality reduction techniques, on five artificial and five RW datasets. The results of the study indicated that the non-linear techniques performed well on artificial data with complex geometrical structures (compared to PCA), but did not clearly indicate that the non-linear techniques outperform PCA on RW tasks. Van der Maaten also stated that these results support other studies in the literature, and that there is no clear indication that non-linear techniques outperform linear techniques on RW datasets.

We therefore focus our attention on PCA as a dimensionality reduction technique. We also prefer PCA as a dimensionality reduction technique, since this is one of the most widely used and well-studied dimensionality reduction techniques [49].

The PCA linear transformation re-aligns the feature axes to the directions of most variation, and thus minimises the variance orthogonal to the projection. Features with small variations in the transformed feature space (small eigenvalues), are regarded as unimportant, and are typically disregarded. The transformed features with largest eigenvalues are typically referred to as the principal components. The PCA transformation also ensures that the features in the transformed feature space are orthogonal, thus ensuring that the data are decorrelated.

It should be noted that the PCA transformation is scale-dependent; thus, the scale of the input variables will influence the directions and weights of the principal compo-

nents. In practice, a first step to PCA is thus to normalise all features in the original feature space to 0 mean and 1 standard deviation. The sample covariance of the normalised data is then calculated, and an eigenvalue transformation is performed on the covariance matrix. The eigenvectors of the transformation provide the weights of the linear transform, while the eigenvalues provide the variance of the transformed features (and thus, possibly, their relative importance).

More formally, a D-dimensional dataset $\mathbf{X}$, can be normalised as

$$\mathbf{Z} = (\mathbf{X} - \mu)\mathbf{\Sigma}^{-1} \tag{3.2}$$

where $\mathbf{Z}$ is the normalised data with features that have 0 mean and 1 standard deviation, μ is the multivariate sample mean of $\mathbf{X}$, and $\mathbf{\Sigma}$ is a diagonal covariance matrix with the sample variances of the D features on the diagonal.

If A is a DxD matrix that defines a linear transformation, then the eigenvalue equation can be written as

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v} \tag{3.3}$$

for some scalar vector λ, where λ gives the eigenvalues of transformation $\mathbf{A}$ and the DxD matrix, $\mathbf{v}$, gives the eigenvectors of transformation $\mathbf{A}$.

The eigenvalue equation can be re-written as

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{X} = 0 \tag{3.4}$$

where I is the DxD identity matrix.

Solving Eq. 3.4 can be viewed as solving a linear system of D equations. The non-trivial solution can thus be obtained by setting the determinant of $(\mathbf{A} - \lambda\mathbf{I})$ equal to zero

$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0. \tag{3.5}$$

This is known as the characteristic equation, and the solution to the characteristic equation provides the solutions to the D eigenvalues in λ. The eigenvectors, v, are then solved by substituting the D eigenvalues into Eq. 3.3.

The PCA linear transformation of the data $\mathbf{X}$ is calculated by substituting the linear transformation matrix A, with the sample covariance Σ into Eq. 3.3. The eigenvalue solution thus gives the linear transformation of the data (eigenvectors) and variances of the features in the transformed space (eigenvalues). This linear transformation is applied to the normalised data, to obtain the transformed data

$$\mathbf{Z}_t = \mathbf{Z}\mathbf{v} \tag{3.6}$$

where $\mathbf{Z}_t$ is the transformed D-dimensional data.

The D-dimensional data, $\mathbf{Z}_t$, can now be reduced to a k-dimensional feature space, $\mathbf{Y}$, by simply finding the features with the k largest eigenvalues, and selecting only these k features from $\mathbf{Z}_t$.

3.6 CONCLUSION

In this chapter we have motivated why it is important in practice for density estimation algorithms to function in high-dimensional spaces, we provided some examples of the non-intuitive nature of high-dimensional data, elaborated on the geometry of feature spaces, explained how KDEs can be used to model the geometry of feature space and explained how linear PCA can be used to perform dimensionality reduction on pattern recognition datasets.

In the next chapter we will turn our attention to kernel bandwidth estimation, and we derive promising bandwidth estimators to make non-parametric density estimation in high-dimensional feature spaces feasible.

Chapter Four

Maximum-likelihood Kernel Bandwidth Estimation

Contents

4.1 Introduction 38

4.2 Kernel Density Estimation 38

4.3 Objective Functions for Optimisation of Bandwidth Selection Procedures 39

4.4 Leave-one-out Likelihood Estimation 40

4.5 Kernel Bandwidth Estimators 40

4.5.1 MLE kernel bandwidth estimation 41

4.5.2 MLL kernel bandwidth estimation 48

4.5.3 ML-LOO kernel bandwidth estimation 54

4.5.4 Summary of ML KDEs computational complexities and parameters to be estimated 57

4.6 Kernel Bandwidth Initialisation 58

4.7 Conclusion 59

4.1 INTRODUCTION

In Chapter 2 we surveyed non-parametric density estimation techniques, and specifically focussed on considerations for kernel estimators. We also briefly surveyed the most promising conventional kernel bandwidth estimators and highlighted some of the short-comings of conventional kernel bandwidth estimation algorithms. The most notable short-coming of conventional kernel estimators is that they were not developed for the purpose of high-dimensional density estimation problems as often encountered in pattern recognition. In Chapter 3 we motivated why it is important in practice for density estimation algorithms to function in high-dimensional spaces, specifically in the context of pattern recognition – and also why such spaces may not admit straightforward extension of low-dimensional approaches.

In this chapter we give a more formal description of kernel density estimation in Section 4.2, define the ML objective function in the context of kernel density estimation in Section 4.3, present the ML LOUT likelihood criterion in Section 4.4, derive ML LOUT kernel bandwidth estimators from first principles in Section 4.5, discuss kernel bandwidth initialisation in Section 4.6 and make concluding remarks in Section 4.7.

4.2 KERNEL DENSITY ESTIMATION

KDEs estimate the PDF of a D-dimensional dataset $\mathbf{X}$, with N *iid* samples $\mathbf{x}_1, ..., \mathbf{x}_N$ with the sum

$$p_{\mathbf{H}}(\mathbf{x}_i) = \frac{1}{N} \sum_{j=1}^{N} K_{\mathbf{H}_j} (\mathbf{x}_i - \mathbf{x}_j | \mathbf{H}_j) \tag{4.1}$$

where $K_{\mathbf{H}_j} (\mathbf{x}_i - \mathbf{x}_j | \mathbf{H}_j)$ is the kernel smoothing function fitted over each data point $\mathbf{x}_j$ with bandwidth matrix $\mathbf{H}_j$. The kernel density estimate is thus non-parametric, since no parametric distribution is imposed on the estimate; instead the estimated distribution is defined by the sum of the kernel functions fitted over the data points.

We see in Eq.4.1 that an appropriate kernel function $K(\cdot)$ must, however, be selected. We discussed the selection of an appropriate kernel function in Section 2.5.1 and noted that the efficiencies of various kernel types in terms of MISE are so close, that the choice of kernel type should rather be based on mathematical properties such as continuity and differentiability, since the estimated PDF will inherit the smoothness properties of the selected kernel. We therefore select the Gaussian kernel based on these properties and will assume a Gaussian kernel for all our derivations.

The bandwidth of each kernel, $\mathbf{H}_j$, determines the amount of smoothing of the density function around each data point $\mathbf{x}_j$. The selection of the appropriate band-

width for each data point is thus crucial for kernel density estimation, since the degree of smoothness of the PDF might vary and the directions of variation might change throughout the feature space.

The estimator must thus be able to (1) model spatial variability in length (to account for variation in density and smoothness), (2) model spatial anisotropy (to account for change in the direction of variation), and (3) function in high dimensional feature spaces. The bandwidth estimation procedure must ensure that all these requirements are met, in order to estimate PDFs of unknown distributions in high-dimensional spaces successfully.

In the next section we will define the ML objective function that can be optimised with respect to kernel bandwidths, $\mathbf{H}_j$, to derive an optimal solution to kernel bandwidths in the ML sense.

4.3 OBJECTIVE FUNCTIONS FOR OPTIMISATION OF BANDWIDTH SELECTION PROCEDURES

The likelihood that the N *iid* samples in dataset $\mathbf{X}$, sampled form an unknown PDF, $\mathrm{p}(\mathbf{x})$, fits the estimated PDF, $p_{\mathbf{H}}(\mathbf{x}_i)$, is defined as

$$L_{\mathbf{H}}(\mathbf{X}) = \prod_{i=1}^{N} p_{\mathbf{H}}(\mathbf{x}_i). \tag{4.2}$$

Since the logarithm function is monotonically increasing, optimising the log-likelihood function is equivalent to optimising the likelihood function. For mathematical and computational convenience, we prefer to make use of the log-likelihood function

$$l_{\mathbf{H}}(\mathbf{X}) = \sum_{i=1}^{N} \log\left[p_{\mathbf{H}}(\mathbf{x}_i)\right]. \tag{4.3}$$

Maximising the log-likelihood in Eq. 4.3 is often referred to as the ML criterion.

Similarly, sample entropy (introduced by Shannon [32]) can be defined as

$$H(\mathbf{X}) = -\frac{1}{N}\sum_{i=1}^{N} \log\left[p_{\mathbf{H}}(\mathbf{x}_i)\right]. \tag{4.4}$$

Substituting Eq. 4.3 into Eq.4.4 shows that the sample entropy can be expressed in terms of the log-likelihood function

$$H(\mathbf{X}) = -\frac{l_{\mathbf{H}}(\mathbf{X})}{N}. \tag{4.5}$$

This indicates that minimising the sample entropy in Eq. 4.4 is equivalent to maximizing the log-likelihood function in Eq. 4.3. This is an important observation, since this result shows that deriving kernel bandwidth estimators that maximise the ML criterion is equivalent to deriving kernel bandwidth estimators that minimise the sample entropy.

4.4 LEAVE-ONE-OUT LIKELIHOOD ESTIMATION

From the definition of a KDE in Eq. 4.1 and the log-likelihood function in Eq. 4.3 the inherent shortcoming of the ML criterion is clear - as the bandwidth for each kernel tends to zero, the likelihood for that data point will tend to infinity. This thus leads to a trivial degenerate solution for the ML objective function. To address this shortcoming the LOUT estimate can be used. Thus, the data point, $\mathbf{x}_i$, for which the density is evaluated, should be left out of the summation in Eq. 4.1. The LOUT formulation of Eq. 4.1 can thus be expressed as

$$p_{\mathbf{H}(-i)}(\mathbf{x}_i) = \frac{1}{N-1} \sum_{j \neq i} K_{\mathbf{H}_j} \left(\mathbf{x}_i - \mathbf{x}_j | \mathbf{H}_j \right) \tag{4.6}$$

The LOUT likelihood function thus leaves out an observation, $\mathbf{x}_i$, from the dataset $\mathbf{X}$ in order to obtain an independent observation that is not part of the dataset. The likelihood of the left out data point, $\mathbf{x}_i$, is then determined by estimating the density, $p_{\mathbf{H}(-i)}(\mathbf{x})$, on the remaining $N-1$ data points.

The LOUT ML objective function for dataset $\mathbf{X}$ is thus obtained by substituting Eq. 4.6 into Eq. 4.3

$$l_{H(-i)}(\mathbf{X}) = \sum_{i=1}^{N} \log \left[p_{\mathbf{H}(-i)}(\mathbf{x}_i) \right] \tag{4.7}$$

Optimising the LOUT ML objective function in Eq. 4.7 with respect to kernel bandwidth matrix will thus prevent the trivial solution of the ML objective function, where $l_{\mathbf{H}}(\mathbf{X}) = \infty$ when $\mathbf{H}_j = 0$.

4.5 KERNEL BANDWIDTH ESTIMATORS

Kernel bandwidth estimation algorithms can be derived from the ML LOUT objective function by either maximising the log-likelihood score or minimising the entropy with respect to the kernel bandwidths. In this section we derive a novel kernel bandwidth estimator, called the minimum leave-one-out entropy (MLE) estimator, that optimises the objective function provided in Eq. 4.7 with respect to the kernel bandwidth matrix

(or equivalently minimise the sample entropy), and makes use of the LOUT formulation for kernel density estimation in Eq. 4.6. We will also show that if certain restrictions are placed on the LOUT formulation in Eq. 4.6, then the MLL and ML-LOO estimators (discussed in Section 1.2 and Section 1.3) can be derived from the same ML framework.

4.5.1 MLE KERNEL BANDWIDTH ESTIMATION

The definition of MLE objective function is obtained by substituting the LOUT KDE, $p_{\mathbf{H}(-i)}(\mathbf{x})$, defined in Eq. 4.6 into the ML objective function defined in Eq. 4.7

$$\begin{aligned} l_{MLE}(\mathbf{X}) &= \sum_{i=1}^{N} \log \left[p_{\mathbf{H}(-i)}(\mathbf{x}_i) \right] \\ &= \sum_{i=1}^{N} \log \left[\frac{1}{N-1} \sum_{j \neq i}^{N} K_{\mathbf{H}_j} \left(\mathbf{x}_i - \mathbf{x}_j | \mathbf{H}_j \right) \right] \end{aligned} \tag{4.8}$$

To estimate the bandwidth $\mathbf{H}_k$ of the kernel centred on data point $\mathbf{x}_k$, from a dataset $\mathbf{X}$ with samples drawn *iid* from an unknown density function $p(\mathbf{x})$, we set the partial derivative of the MLE objective function (given in Eq. 4.8) with respect to the bandwidths $\mathbf{H}_k$ equal to zero and solve for $\mathbf{H}_k$. The partial derivative of the MLE objective function with respect to the bandwidth $\mathbf{H}_k$ can be expressed as

$$\begin{aligned} \frac{\partial}{\partial \mathbf{H}_k} \left(l_{MLE}(\mathbf{X}) \right) &= \frac{\partial}{\partial \mathbf{H}_k} \left(\sum_{i=1}^{N} \log \left[p_{\mathbf{H}(-i)}(\mathbf{x}_i) \right] \right) \\ &= \sum_{i=1}^{N} \frac{\partial}{\partial \mathbf{H}_k} \left(\log \left[p_{\mathbf{H}(-i)}(\mathbf{x}_i) \right] \right) \end{aligned} \tag{4.9}$$

and the partial derivative of the log term as

$$\begin{aligned} \frac{\partial}{\partial \mathbf{H}_k} \left(\log \left[p_{\mathbf{H}(-i)}(\mathbf{x}_i) \right] \right) &= \frac{\frac{\partial}{\partial \mathbf{H}_k} \left(p_{\mathbf{H}(-i)}(\mathbf{x}_i) \right)}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)} \\ &= \frac{\frac{\partial}{\partial \mathbf{H}_k} \left(\frac{1}{N-1} \sum_{j \neq i} K_{\mathbf{H}_j} (\mathbf{x}_i - \mathbf{x}_j | \mathbf{H}_j) \right)}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)} \end{aligned} \tag{4.10}$$

Since partial derivative with respect to $\mathbf{H}_k$ will only be non-zero for j=k, Eq. 4.10 can be simplified to

$$\frac{\partial}{\partial \mathbf{H}_k} \left(\log \left[p_{\mathbf{H}(-i)}(\mathbf{x}_i) \right] \right) = \frac{\frac{1}{N-1} \frac{\partial}{\partial \mathbf{H}_k} \left[K_{\mathbf{H}_k} \left(\mathbf{x}_i - x_k | \mathbf{H}_k \right) \right]}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)} \tag{4.11}$$

If we substitute the partial derivative in Eq. 4.11 into Eq. 4.9, we obtain the partial derivative of the MLE objective function

$$\frac{d}{d\mathbf{H}_k} \left(l_{MLE}(\mathbf{X}) \right) = \frac{1}{N-1} \sum_{i=1}^{N} \frac{\frac{\partial}{\partial \mathbf{H}_k} \left[K_{\mathbf{H}_k} \left(\mathbf{x}_i - x_k | \mathbf{H}_k \right) \right]}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)} \tag{4.12}$$

We can now obtain the MLE bandwidth estimate $\mathbf{H}_k$ by setting this derivative to zero and solving for $\mathbf{H}_k$. We note, however, that the derivative in Eq. 4.12 is dependent on the derivative of the kernel function. We will therefore provide the derivatives of the

univariate Gaussian kernel, multivariate Gaussian kernel with full-covariance matrix and multivariate Gaussian kernel with diagonal covariance matrix and will substitute these derivatives into Eq. 4.12 to obtain the MLE solutions for $\mathbf{H}_k$.

4.5.1.1 UNIVARIATE GAUSSIAN KERNEL

If we assume a univariate Gaussian kernel, the kernel function can be expressed as the standard univariate normal density function and the derivative of the kernel with respect to h_k as

$$\frac{d}{dh_k}\left[K_{h_k}\left(\frac{x_i - x_k}{h_k}\right)\right] = \frac{1}{h_k^2}\left[\frac{K\left(\frac{x_i - x_k}{h_k}\right)(x_i - x_k)^2}{h_k^2} - K\left(\frac{x_i - x_k}{h_k}\right)\right] \tag{4.13}$$

We now substitute Eq. 4.13 into the MLE objective function in Eq. 4.12 to obtain

$$\frac{d}{dh_k}\left(l_{MLE}(X)\right) = \frac{1}{(N-1)h_k^2}\sum_{i=1}^{N}\frac{\left[\frac{1}{h_k^2}K\left(\frac{x_i - x_k}{h_k}\right)(x_i - x_k)^2 - K\left(\frac{x_i - x_k}{h_k}\right)\right]}{p_{H(-i)}(x_i)} \tag{4.14}$$

We set the derivative in Eq. 4.14 to zero

$$\frac{1}{h_k^2}\sum_{i=1}^{N}\frac{K\left(\frac{x_i - x_k}{h_k}\right)(x_i - x_k)^2}{p_{H(-i)}(x_i)} = \sum_{i=1}^{N}\frac{K\left(\frac{x_i - x_k}{h_k}\right)}{p_{H(-i)}(x_i)} \tag{4.15}$$

and solve for h_k^2

$$h_k^2 = \frac{\sum_{i=1}^{N}\frac{K\left(\frac{x_i - x_k}{h_k}\right)(x_i - x_k)^2}{p_{H(-i)}(x_i)}}{\sum_{i=1}^{N}\frac{K\left(\frac{x_i - x_k}{h_k}\right)}{p_{H(-i)}(x_i)}} \tag{4.16}$$

The calculation of h_k^2 in Eq. 4.16 requires N evaluations of the leave-one-out likelihood estimate, $p_{H(-i)}(x_i)$, where the calculation of $p_{H(-i)}(x_i)$ requires $N-1$ evaluations of the univariate Gaussian kernel function. This yields a time complexity of order $O(N^2)$ for the calculation of Eq. 4.16 and since h_k^2 must be calculated for each of the N training samples, this yields a time complexity of $O(N^3)$ for the univariate MLE Gaussian estimator.

The parameters to estimate are the N bandwidths, h_k^2, and thus scale linearly with the number of data points.

4.5.1.2 MULTIVARIATE GAUSSIAN KERNEL (FULL COVARIANCE MATRIX)

The multivariate Gaussian density function with full covariance matrix $\mathbf{H}_k$ can be expressed as

$$K_{\mathbf{H}_k}\left(\mathbf{x}_i - \mathbf{x}_k | \mathbf{H}_k\right) = \frac{1}{(2\pi)^{D/2}\left|\mathbf{H}_k\right|^{\frac{1}{2}}}e^{-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_k)^T\mathbf{H}_k^{-1}(\mathbf{x}_i - \mathbf{x}_k)} \tag{4.17}$$

The partial derivative of this density function with respect to full covariance matrix $\mathbf{H}_k$ can be derived as follows. First, we set

$$a(\mathbf{H}_k) = \frac{|\mathbf{H}_k|^{\frac{1}{2}}}{(2\pi)^{D/2}} \tag{4.18}$$

and

$$b(\mathbf{H}_k) = e^{c(\mathbf{H}_k)} \tag{4.19}$$

where

$$c(\mathbf{H}_k) = -\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_k)^T \mathbf{H}_k^{-1}(\mathbf{x}_i - \mathbf{x}_k) \tag{4.20}$$

We then derive the partial derivative of $a(\mathbf{H}_k)$ by means of the chain rule

$$\frac{\partial}{\partial \mathbf{H}_k} a(\mathbf{H}_k) = \frac{-\frac{1}{2}\frac{\partial}{\partial \mathbf{H}_k}(|\mathbf{H}_k|)|\mathbf{H}_k|^{-\frac{3}{2}}}{(2\pi)^{D/2}} \tag{4.21}$$

If we substitute the matrix derivative identity [50]

$$\begin{aligned} \tfrac{\partial}{\partial \mathbf{X}}(|\mathbf{X}|) &= |\mathbf{X}|(X^{-1})^T \\ &= |\mathbf{X}|\mathbf{X}^{-1} \end{aligned} \tag{4.22}$$

where the second step uses the matrix property $(\mathbf{X}^{-1})^T = (\mathbf{X}^T)^{-1}$ and assumes that $|\mathbf{X}|$is symmetric and therefore $|\mathbf{X}|$ is equal to its transpose, thus $(X^{-1})^T = \mathbf{X}^{-1}$, we obtain

$$\frac{\partial}{\partial \mathbf{H}_k} a(\mathbf{H}_k) = \frac{-\frac{1}{2}|\mathbf{H}_k|^{-\frac{1}{2}}\mathbf{H}_k^{-1}}{(2\pi)^{D/2}} \tag{4.23}$$

The partial derive of $b(\mathbf{H}_k)$is given by

$$\frac{\partial}{\partial \mathbf{H}_k} b(\mathbf{H}_k) = c(\mathbf{H}_k)e^{c(\mathbf{H}_k)} \tag{4.24}$$

and the partial derivative of $c(\mathbf{H}_k)$is

$$\frac{\partial}{\partial \mathbf{H}_k} c(\mathbf{H}_k) = \frac{1}{2}\left[\mathbf{H}_k^{-1}(\mathbf{x}_i - \mathbf{x}_k)(\mathbf{x}_i - \mathbf{x}_k)^T\mathbf{H}_k^{-1}\right] \tag{4.25}$$

where we make use of the matrix derivative identity

$$\begin{aligned} \tfrac{\partial}{\partial \mathbf{X}}(\mathbf{a}\mathbf{X}^{-1}\mathbf{b}) &= -\mathbf{X}^{-T}\mathbf{a}\mathbf{b}\mathbf{X}^{-T} \\ &= \mathbf{X}^{-1}\mathbf{a}\mathbf{b}\mathbf{X}^{-1} \end{aligned} \tag{4.26}$$

and the second step assumes again that $\mathbf{X}$is symmetric.

We now substitute these derivatives into

$$\frac{\partial}{\partial \mathbf{H}_k} K_{\mathbf{H}_k}(\mathbf{x}_i - \mathbf{x}_k|\mathbf{H}_k) = a(\mathbf{H}_k)\frac{\partial}{\partial \mathbf{H}_k} b(\mathbf{H}_k) + b(\mathbf{H}_k)\frac{\partial}{\partial \mathbf{H}_k} a(\mathbf{H}_k) \tag{4.27}$$

to obtain the partial derivative of the multivariate Gaussian density function with full covariance matrix $\mathbf{H}_k$

$$\frac{\partial}{\partial \mathbf{H}_k} K_{\mathbf{H}_k}(\mathbf{x}_i - \mathbf{x}_k|\mathbf{H}_k) = \frac{1}{2}\mathbf{H}_k^{-2}(\mathbf{x}_i - \mathbf{x}_k)(\mathbf{x}_i - \mathbf{x}_k)^T K_{\mathbf{H}_k}(\mathbf{x}_i - \mathbf{x}_k|\mathbf{H}_k) - \frac{1}{2}\mathbf{H}_k^{-1} K_{\mathbf{H}_k}(\mathbf{x}_i - \mathbf{x}_k|\mathbf{H}_k) \tag{4.28}$$

We obtain the partial derivative of the MLE objective function by substituting Eq. 4.28 into the MLE objective function in Eq. 4.12

$$\frac{d}{d\mathbf{H}_k}\left(l_{MLE}(\mathbf{X})\right) = \frac{1}{N-1}\sum_{i=1}^{N}\left[\frac{\frac{1}{2}\mathbf{H}_k^{-2}(\mathbf{x}_i-\mathbf{x}_k)(\mathbf{x}_i-\mathbf{x}_k)^T K_{\mathbf{H}_k}(\mathbf{x}_i-\mathbf{x}_k|\mathbf{H}_k) - \frac{1}{2}\mathbf{H}_k^{-1} K_{\mathbf{H}_k}(\mathbf{x}_i-\mathbf{x}_k|\mathbf{H}_k)}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)}\right] \tag{4.29}$$

We then set the partial derivative of the MLE objective function in Eq. 4.29 to zero

$$\sum_{i=1}^{N}\frac{(\mathbf{x}_i - \mathbf{x}_k)(\mathbf{x}_i - \mathbf{x}_k)^T K_{\mathbf{H}_k}(\mathbf{x}_i - \mathbf{x}_k|\mathbf{H}_k)}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)} = \mathbf{H}_k \sum_{i=1}^{N}\frac{K_{\mathbf{H}_k}(\mathbf{x}_i - \mathbf{x}_k|\mathbf{H}_k)}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)} \tag{4.30}$$

and solve for $\mathbf{H}_k$

$$\mathbf{H}_k = \frac{\sum_{i=1}^{N}\frac{(\mathbf{x}_i-\mathbf{x}_k)(\mathbf{x}_i-\mathbf{x}_k)^T K_{\mathbf{H}_k}(\mathbf{x}_i-\mathbf{x}_k|\mathbf{H}_k)}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)}}{\sum_{i=1}^{N}\frac{K_{\mathbf{H}_k}(\mathbf{x}_i-\mathbf{x}_k|\mathbf{H}_k)}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)}} \tag{4.31}$$

where $\mathbf{H}_k$ is the estimate of the MLE full covariance bandwidth matrix for the kernel centred on data point $\mathbf{x}_k$.

The time complexity of evaluating the multivariate Gaussian kernel, $K_{H_k}(x_i - x_k|H_k)$, with full covariance matrix, H_k, is of order $O(D^3)$ if Gauss-Jordan elimination is used to calculate the inverse of H_k and LU decomposition is used to calculate the determinant of H_k, since both these algorithms have a time complexity of order $O(D^3)$. Therefore, the time complexity of evaluating the LOUT likelihood function, $p_{H(-i)}(x_i)$, in Eq. 4.31 is of order $O(ND^3)$ since $N-1$ calculations of the kernel function are required. The calculation of H_k in Eq. 4.31 requires N evaluations of $p_{H(-i)}(x_i)$, which yields a time complexity of order $O(N^2D^3)$. Since H_k must be calculated for each of the N training data points, the time complexity of the MLE full covariance estimator is of order $O(N^3D^3)$.

The parameters to estimate are the $D(D+1)/2$ parameters for each of the N symmetric full covariance matrices; therefore the number of parameters to estimate is $ND(D+1)/2$. The number of parameters thus scales linearly with number of training points and quadratically with the dimensionality of the dataset.

4.5.1.3 MULTIVARIATE GAUSSIAN KERNEL (DIAGONAL COVARIANCE MATRIX)

The multivariate Gaussian density function with diagonal covariance matrix $\mathbf{H}_j$ can be expressed as the product of univariate Gaussian density functions

$$K_{\mathbf{H}_j}\left(\mathbf{x}_i - \mathbf{x}_j | \mathbf{H}_j\right) = \prod_{p=1}^{D} K_{h_{jp}}\left(\frac{x_{ip} - x_{jp}}{h_{jp}}\right) \tag{4.32}$$

where x_{ip} is the value of data point $\mathbf{x}_i$ in dimension p, x_{jp} is the value of data point $\mathbf{x}_j$ in dimension p and h_{jp} is the bandwidth of the univariate Gaussian kernel $K_{h_{jp}}(\cdot)$ centred on data point $\mathbf{x}_j$ in dimension p and $\mathbf{H}_j$ is the diagonal bandwidth matrix such that $\mathbf{H}_{\mathbf{j(p,p)}} = h_{jp}^2$.

The partial derivative of the multivariate Gaussian expression in Eq. 4.32 with respect to the diagonal bandwidth matrix $\mathbf{H}_j$ in dimension d can be derived by means of the product rule

$$\begin{aligned}\frac{\partial}{\partial h_{jd}}\left[K_{\mathbf{H}_j}\left(\mathbf{x}_i - \mathbf{x}_j | \mathbf{H}_j\right)\right] &= \frac{\partial}{\partial h_{jd}}\left[\prod_{p=1}^{D} K_{h_{jd}}\left(\frac{x_{id}-x_{jd}}{h_{jd}}\right)\right] \\ &= \frac{1}{h_{jd}^2}\left[\frac{K_{h_{jd}}\left(\frac{x_{id}-x_{jd}}{h_{jd}}\right)(x_{id}-x_{jd})^2}{h_{jd}^2} - K_{h_{jd}}\left(\frac{x_{id}-x_{jd}}{h_{jd}}\right)\right]\prod_{p\neq d} K_{h_{jp}}\left(\frac{x_{ip}-x_{jp}}{h_{jp}}\right)\end{aligned} \tag{4.33}$$

If we substitute the derivative of the kernel function in Eq. 4.33 into Eq. 4.12, then the partial derivative of the MLE objective function with respect to the bandwidth $\mathbf{H}_k$ in dimension d is

$$\frac{d}{dh_{kd}}\left(l_{MLE}(\mathbf{X})\right) = \frac{1}{(N-1)h_{kd}^2}\sum_{i=1}^{N}\left[\frac{K_{h_{kd}}\left(\frac{x_{id}-x_{kd}}{h_{kd}}\right)(x_{id}-x_{kd})^2}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)h_{kd}^2} - \frac{K_{h_{kd}}\left(\frac{x_{id}-x_{kd}}{h_{kd}}\right)}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)}\right]\prod_{p\neq d} K_{h_{kp}}\left(\frac{x_{ip}-x_{kp}}{h_{kp}}\right) \tag{4.34}$$

where h_{kd} is the bandwidth of the univariate Gaussian kernel $K_{h_{kd}}(\cdot)$ centred on data point $\mathbf{x}_k$ in dimension d.

If we set the derivative in Eq. 4.34 to zero and solve for h_{kd}^2 we obtain

$$h_{kd}^2 = \frac{\sum_{i=1}^{N}\frac{K_{h_{kd}}\left(\frac{x_{id}-x_{kd}}{h_{kd}}\right)(x_{id}-x_{kd})^2}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)}\prod_{p\neq d} K_{h_{kp}}\left(\frac{x_{ip}-x_{kp}}{h_{kp}}\right)}{\sum_{i=1}^{N}\frac{K_{h_{kd}}\left(\frac{x_{id}-x_{kd}}{h_{kd}}\right)}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)}\prod_{p\neq d} K_{h_{kp}}\left(\frac{x_{ip}-x_{kp}}{h_{kp}}\right)} \tag{4.35}$$

If we substitute the definition of the multivariate Gaussian density function in Eq. 4.32 into Eq. 4.35, we finally obtain

$$H_{k(d,d)} = \frac{\sum_{i=1}^{N}\frac{K_{\mathbf{H}_k}(\mathbf{x}_i-\mathbf{x}_k|\mathbf{H}_k)(x_{id}-x_{kd})^2}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)}}{\sum_{i=1}^{N}\frac{K_{\mathbf{H}_k}(\mathbf{x}_i-\mathbf{x}_k|\mathbf{H}_k)}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)}} \tag{4.36}$$

where $\mathbf{H}_k$ is the diagonal bandwidth matrix such that $\mathbf{H}_{k(d,d)} = h_{kd}^2$.

If we compare the bandwidth solutions in Eq. 4.31 and Eq. 4.36, we see that the outer product $(\mathbf{x}_i - \mathbf{x}_k)(\mathbf{x}_i - \mathbf{x}_k)^T$ can simply be replaced with the squared distance measure $(x_{id} - x_{kd})^2$ for each dimension d, instead of a DxD matrix. Note that assuming diagonal kernel bandwidth matrices is not the same as assuming independence between variables. Kernel estimators with diagonal bandwidth matrices are capable of modelling correlation to some degree, since they are centred on data points and if data points vary together between dimensions, the kernel density function will capture the co-variance. If, however, only a single Gaussian is fitted on the data, then a diagonal covariance matrix will assume independence between variables – this is, however, not applicable to KDEs.

The time complexity of evaluating the multivariate Gaussian kernel, $K_{H_k}(x_i - x_k|H_k)$, with diagonal covariance matrix, H_k, is of order $O(D)$ since the inverse of the covariance matrix is simply the inverse of each of the diagonal elements and the determinant is simply the product of the diagonal elements. Eq. 4.36 requires N calculations of the leave-one-out likelihood estimate, $p_{H(-i)}(x_i)$, which in turn requires $N-1$ calculations of the multivariate Gaussian kernel function. This yields a time complexity of order $O(N^2D)$ for Eq. 4.36. Since H_k must be calculated for the N training data points, the time complexity of the diagonal MLE estimator is of order $O(N^3D)$.

The parameters to estimate are the D parameters for each of the N diagonal covariance matrices; therefore the number of parameters to estimate is ND. The number of parameters thus scales linearly with the number of data points in the training set and with the dimensionality of the dataset.

4.5.1.4 GLOBAL MLE ESTIMATOR

We also derive an MLE kernel bandwidth estimator that estimates an identical diagonal bandwidth matrix for all kernels (as opposed to a unique bandwidth matrix per kernel as in the case of the standard MLE estimator). We call this estimator the global MLE estimator, since it estimates a single diagonal bandwidth matrix used globally.

More formally, the LOUT KDE estimate for the global MLE estimator is defined

$$p_{\mathbf{H}_g(-i)}(\mathbf{x}_i) = \frac{1}{N-1}\sum_{j \neq i}^{N} K_{\mathbf{H}_g}(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}_g) \tag{4.37}$$

where $\mathbf{H}_g$ is the diagonal bandwidth matrix that is identical for all kernels. The global

MLE objective function can thus be written as

$$\begin{aligned} l_{MLE(g)}(\mathbf{X}) &= \sum_{i=1}^{N} \log \left[p_{\mathbf{H}_g(-i)}(\mathbf{x}_i) \right] \\ &= \sum_{i=1}^{N} \log \left[\frac{1}{N-1} \sum_{j \neq i}^{N} K_{\mathbf{H}_g} \left(\mathbf{x}_i - \mathbf{x}_j | \mathbf{H}_g \right) \right] \end{aligned} \tag{4.38}$$

and the partial derivative of the global MLE objective function with respect to the diagonal bandwidth matrix $\mathbf{H}_g$ can be expressed as

$$\begin{aligned} \frac{\partial}{\partial \mathbf{H}_g} \left(l_{MLE(g)}(\mathbf{X}) \right) &= \frac{\partial}{\partial \mathbf{H}_g} \left(\sum_{i=1}^{N} \log \left[p_{\mathbf{H}_g(-i)}(\mathbf{x}_i) \right] \right) \\ &= \sum_{i=1}^{N} \frac{\partial}{\partial \mathbf{H}_g} \left(\log \left[p_{\mathbf{H}_g(-i)}(\mathbf{x}_i) \right] \right) \end{aligned} \tag{4.39}$$

where the partial derivative of the log term is

$$\begin{aligned} \frac{\partial}{\partial \mathbf{H}_g} \left(\log \left[p_{\mathbf{H}_g(-i)}(\mathbf{x}_i) \right] \right) &= \frac{\frac{\partial}{\partial \mathbf{H}_g} \left(p_{\mathbf{H}(-i)}(\mathbf{x}_i) \right)}{p_{\mathbf{H}_g(-i)}(\mathbf{x}_i)} \\ &= \frac{\frac{\partial}{\partial \mathbf{H}_g} \left(\frac{1}{N-1} \sum_{j \neq i} K_{\mathbf{H}_g} (\mathbf{x}_i - \mathbf{x}_j | \mathbf{H}_g) \right)}{p_{\mathbf{H}_g(-i)}(\mathbf{x}_i)} \end{aligned} \tag{4.40}$$

We note that we cannot remove the summation term in Eq. 4.40 as in the case of the MLE estimator in Eq. 4.11, since the derivative is with respect to $\mathbf{H}_g$ instead of $\mathbf{H}_j$, and will not necessarily be zero for all summands where $j \neq k$.

If we substitute the partial derivative in Eq. 4.40 into the partial derivative of the global MLE objective function in Eq. 4.39 we obtain

$$\frac{d}{d\mathbf{H}_g} \left(l_{MLE(g)}(\mathbf{X}) \right) = \sum_{i=1}^{N} \frac{\frac{1}{N-1} \sum_{j \neq i} \frac{\partial}{\partial \mathbf{H}_g} \left[K_{\mathbf{H}_g} \left(\mathbf{x}_i - \mathbf{x}_j | \mathbf{H}_g \right) \right]}{p_{\mathbf{H}_g(-i)}(\mathbf{x}_i)} \tag{4.41}$$

The partial derivative of the multivariate Gaussian distribution with diagonal bandwidth matrix $\mathbf{H}_g$ in dimension d can be derived (similar to Eq. 4.33) as

$$\frac{\partial}{\partial h_d} \left[K_{\mathbf{H}_g} \left(\mathbf{x}_i - \mathbf{x}_j | \mathbf{H}_g \right) \right] = \frac{1}{h_d^2} \left[\frac{K_{h_d} \left(\frac{x_{id} - x_{jd}}{h_d} \right) (x_{id} - x_{jd})^2}{h_d^2} - K_{h_d} \left(\frac{x_{id} - x_{jd}}{h_d} \right) \right] \prod_{p \neq d} K_{h_p} \left(\frac{x_{ip} - x_{jp}}{h_p} \right) \tag{4.42}$$

where h_d is the kernel bandwidth in dimension d used for all kernels.

If we substitute this derivative into the global MLE objective function in Eq. 4.41 then set the objective function to zero and solve for h_d^2 we obtain

$$h_d^2 = \frac{\sum_{i=1}^{N} \frac{\sum_{j \neq i} K_{h_d} \left(\frac{x_{id} - x_{jd}}{h_d} \right) (x_{id} - x_{jd})^2 \prod_{p \neq d} K_{h_p} \left(\frac{x_{ip} - x_{jp}}{h_p} \right)}{p_{\mathbf{H}_g(-i)}(\mathbf{x}_i)}}{\sum_{i=1}^{N} \frac{\sum_{j \neq i} K_{h_d} \left(\frac{x_{id} - x_{jd}}{h_d} \right) \prod_{p \neq d} K_{h_p} \left(\frac{x_{ip} - x_{jp}}{h_p} \right)}{p_{\mathbf{H}_g(-i)}(\mathbf{x}_i)}} \tag{4.43}$$

If we substitute the definition of the multivariate Gaussian density function in Eq. 4.32 into Eq. 4.43, we obtain the solution of the d^{th} diagonal component of the global MLE

diagonal bandwidth matrix

$$\mathbf{H}_{g(d,d)} = \frac{\sum_{i=1}^{N} \frac{\sum_{j \neq i} K_{\mathbf{H}_g}\left(x_{id} - x_{jd} | \mathbf{H}_g\right)(x_{id} - x_{jd})^2}{p_{\mathbf{H}_g(-i)}(\mathbf{x}_i)}}{\sum_{i=1}^{N} \frac{\sum_{j \neq i} K_{\mathbf{H}_g}\left(x_{id} - x_{jd} | \mathbf{H}_g\right)}{p_{\mathbf{H}_g(-i)}(\mathbf{x}_i)}} \tag{4.44}$$

The time complexity of evaluating the multivariate Gaussian kernel, $K_{H_g}(x_{id} - x_{jd}|H_g)$, with diagonal covariance matrix, H_g, is of order $O(D)$. Therefore, time complexity of evaluating the LOUT likelihood function, $p_{H_g(-i)}(x_i)$, in Eq. 4.44 is of order $O(ND)$ since $N-1$ calculations of the kernel function are required. The calculation of $H_{g(d,d)}$ in Eq. 4.44 requires N evaluations of $p_{H_g(-i)}(x_i)$, which yields a time complexity of order $O(N^2D)$.

The parameters to estimate are the D diagonal elements of H_g, the number of parameters thus scales linearly with the dimensionality and is not influenced by the number of training data points, N.

4.5.2 MLL KERNEL BANDWIDTH ESTIMATION

We have thus far derived the MLE bandwidth estimator from the ML LOUT framework and in this section we will show that the MLL bandwidth estimation algorithm that was recently introduced by Barnard [17], can also be derived from the same ML framework. We perform the derivations of the MLL estimator since the full proof of this algorithm was not published and since this will allow us to compare the MLL estimator to the other ML estimators derived in this chapter in a common theoretical framework.

A key insight observed by [51, 17] is that if the density function changes relatively slowly throughout feature space, the optimal kernel bandwidths will also change slowly throughout feature space. If it assumed that the density function changes slowly throughout feature space, a simplification can be made by assuming that the bandwidths of kernels in the neighbourhood of a kernel, are the same as that of the kernel. Thus, when the bandwidth $\mathbf{H}_i$ is being estimated, it is assumed that all other bandwidths are equal to $\mathbf{H}_i$. This reduces the complexity of the bandwidth optimisation problem significantly, since the LOUT kernel density estimate in Eq. 4.6 can be reduced to

$$p_{\mathbf{H}_i(-i)}(\mathbf{x}_i) = \frac{1}{N-1} \sum_{j \neq i} K_{\mathbf{H}_i}(\mathbf{x}_i - \mathbf{x}_j | \mathbf{H}_i) \tag{4.45}$$

This implies that the optimisation of the ML objective function in Eq. 4.7 with respect to the bandwidths, will not require partial derivatives with respect to the $N-1$ bandwidths $\mathbf{H}_j$, and will require only the derivative with respect to the bandwidth being estimated $\mathbf{H}_i$.

If an exponential kernel, such as the Gaussian kernel, is used the value of $K_{\mathbf{H}_i}(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}_i)$ will decrease exponentially as the distance $\|\mathbf{x}_i - \mathbf{x}_j\|$ increases. Thus the contributions of the kernels centred at locations $\mathbf{x}_j$ will decrease exponentially as the distance $\|\mathbf{x}_i - \mathbf{x}_j\|$ increases. This implies that the estimate of the density for data point $\mathbf{x}_i$, given in Eq. 4.45 will not be significantly influenced by data points that are not in the neighbourhood of $\mathbf{x}_i$. Given this observation, the assumption that all kernels have bandwidth $\mathbf{H}_i$ when estimating $\mathbf{H}_i$ in Eq. 4.45, might be reasonable if the density changes slowly within the neighbourhood of $\mathbf{x}_i$.

If, however, the density varies quickly throughout feature space, the effect of this assumption is not negligible. The kernels of nearby data points will have significantly different bandwidths from $\mathbf{H}_i$ (since the density varies quickly), and since the data points in the neighbourhood of $\mathbf{x}_i$ are not far enough such that $K_{\mathbf{H}_i}(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}_i) \approx 0$, their contributions to the sum in Eq. 4.45 will not be negligible.

We will refer to the kernel bandwidth estimator based on $p_{\mathbf{H}(-i)}(x)$ as the MLL estimator, since this formulation is based on the MLL estimator introduced by [17]. We thus define the ML objective function for the MLL estimator by substituting Eq. 4.45 into Eq. 4.7

$$\begin{aligned} l_{MLL}(\mathbf{X}) &= \sum_{i=1}^{N} \log\left[p_{\mathbf{H}_i(-i)}(\mathbf{x}_i)\right] \\ &= \sum_{i=1}^{N} \log\left[\frac{1}{N-1}\sum_{j\neq i} K_{\mathbf{H}_i}(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}_i)\right] \end{aligned} \tag{4.46}$$

The main distinction between the MLL and MLE algorithms is the difference in formulations of the LOUT KDE, which is caused by the assumption of the MLL estimator that the density function changes slowly throughout feature space.

We estimate the kernel bandwidth matrix $\mathbf{H}_k$ of the kernel centred on data point $\mathbf{x}_k$, from a dataset $\mathbf{X}$ drawn *iid* from an unknown density function $p(\mathbf{x})$ by optimising the MLL objective function in Eq. 4.46 with respect to the bandwidth $\mathbf{H}_k$. The partial derivative of the MLL objective function with respect to the bandwidth $\mathbf{H}_k$ can be expressed as

$$\begin{aligned} \frac{\partial}{\partial \mathbf{H}_k}\left(l_{MLL}(\mathbf{X})\right) &= \frac{\partial}{\partial \mathbf{H}_k}\left(\sum_{i=1}^{N} \log\left[p_{\mathbf{H}_i(-i)}(\mathbf{x}_i)\right]\right) \\ &= \sum_{i=1}^{N} \frac{\partial}{\partial \mathbf{H}_k}\left(\log\left[p_{\mathbf{H}_i(-i)}(\mathbf{x}_i)\right]\right) \end{aligned} \tag{4.47}$$

where the derivative of the log term is

$$\begin{aligned} \frac{\partial}{\partial \mathbf{H}_k}\left(\log\left[p_{\mathbf{H}_i(-i)}(\mathbf{x}_i)\right]\right) &= \frac{\frac{\partial}{\partial \mathbf{H}_k}\left(p_{\mathbf{H}_i(-i)}(\mathbf{x}_i)\right)}{p_{\mathbf{H}_i(-i)}(\mathbf{x}_i)} \\ &= \frac{\frac{1}{N-1}\sum_{j\neq i}\frac{\partial}{\partial \mathbf{H}_k} K_{\mathbf{H}_i}(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}_i)}{p_{\mathbf{H}_i(-i)}(\mathbf{x}_i)} \end{aligned} \tag{4.48}$$

Since the partial derivative in the summation term in Eq. 4.47 will be zero where $i \neq k$

the summation term can be dropped and we can substitute Eq. 4.48 to obtain

$$\frac{\partial}{\partial \mathbf{H}_i}\left(l_{MLL}(\mathbf{X})\right) = \frac{\frac{1}{N-1}\sum_{j\neq i}\frac{\partial}{\partial \mathbf{H}_i}K_{\mathbf{H}_i}\left(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}_i\right)}{p_{\mathbf{H}_i(-i)}(\mathbf{x}_i)} \tag{4.49}$$

The optimal MLL bandwidth can now be obtained by setting the derivative of the MLL objective function in Eq. 4.49 to zero and solving for $\mathbf{H}_i$. We will first solve this derivative for the univariate Gaussian case, and then extend the formulation to the multivariate Gaussian case for both a full and diagonal covariance matrix.

4.5.2.1 UNIVARIATE GAUSSIAN KERNEL

If we assume a univariate Gaussian kernel, the kernel function can be expressed as the standard univariate normal density function

$$K\left(\frac{x_i - x_j}{h_i}\right) = \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x_i - x_j}{h_i}\right)^2} \tag{4.50}$$

and its derivative with respect to h_i

$$\frac{d}{dh_i}\left[K\left(\frac{x_i - x_j}{h_i}\right)\right] = \frac{(x_i - x_j)^2}{h_i^3}K\left(\frac{x_i - x_j}{h_i}\right) \tag{4.51}$$

The univariate Gaussian density function can be expressed as

$$K_{h_i}\left(\frac{x_i - x_j}{h_i}\right) = \frac{1}{h_i}K\left(\frac{x_i - x_j}{h_i}\right) \tag{4.52}$$

and its derivative with respect to h_i

$$\frac{d}{dh_i}\left[K_{h_i}\left(\frac{x_i - x_j}{h_i}\right)\right] = \frac{K\left(\frac{x_i - x_j}{h_i}\right)(x_i - x_j)^2}{h_i^4} - \frac{K\left(\frac{x_i - x_j}{h_i}\right)}{h_i^2} \tag{4.53}$$

We obtain the numerator in Eq. 4.49 by substituting Eq. 4.53 into Eq. 4.45 (for the univariate case)

$$\frac{d}{dh_i}\left(p_{h_i(-i)}(x_i)\right) = \frac{1}{N-1}\left[\sum_{j\neq i}\frac{K\left(\frac{x_i - x_j}{h_i}\right)(x_i - x_j)^2}{h_i^4} - \sum_{j\neq i}\frac{K\left(\frac{x_i - x_j}{h_i}\right)}{h_i^2}\right] \tag{4.54}$$

We now substitute Eq. 4.54 into Eq. 4.49 to obtain the derivative of the MLL objective function, and set the derivative to zero to solve for h_i^2

$$h_i^2 = \frac{\sum_{j\neq i}K\left(\frac{x_i - x_j}{h_i}\right)(x_i - x_j)^2}{\sum_{j\neq i}K\left(\frac{x_i - x_j}{h_i}\right)} \tag{4.55}$$

The solution to the optimal MLL bandwidth given in Eq. 4.55 can be obtained by either making use of a conventional optimisation algorithm e.g. a gradient based optimiser, or h_i can be solved with a more direct approach where the right-hand side is initialised with a given bandwidth, the left hand side is updated, and the updated bandwidth is then substituted into the right-hand side again. This process can be performed iteratively until the bandwidth converges.

The calculation of h_i^2 in Eq. 4.55 requires N-1 evaluations of the univariate Gaussian kernel, $K\left(\frac{x_i - x_j}{h_i}\right)$, which yields a time complexity of order $O(N)$ for Eq. 4.55. Since h_i^2 must be calculated N times, this yields a time complexity of order $O(N^2)$ for the MLL univariate Gaussian estimator.

The parameters to estimate are simply the N bandwidth values, and therefore the number of parameters scales linearly with the number of training data points, N.

We note that this expression for the optimal MLE bandwidth is very similar to the MLL bandwidth expression in Eq. 4.55, and that we can express both equations more generally as

$$h_k^2 = \frac{\sum_{i=1}^{N} w_{ik}(x_i - x_k)^2}{\sum_{i=1}^{N} w_{ik}}, \tag{4.56}$$

where the weight function, w_{ik}, for the MLL bandwidth estimator is

$$w_{ik(MLL)} = K\left(\frac{x_i - x_k}{h_k}\right), \tag{4.57}$$

and for the MLE bandwidth estimator is

$$w_{ik(MLE)} = \frac{K\left(\frac{x_i - x_k}{h_k}\right)}{p_{H(-i)}(x_i)}, \tag{4.58}$$

From this formulation we see that the MLE weight function, in Eq. 4.58, has a normalisation factor, $p_{H_i(-i)}(x_i)$, at each iteration i, where the value of $p_{H_i(-i)}(x_i)$ is the MLE LOUT likelihood score for data point x_i. Thus, as the likelihood of data point x_i increases, the relative weight of x_i decreases, and as the likelihood of x_i decreases, the relative weight of x_i increases. If we apply this insight to Eq. 4.56, it is clear that when the bandwidth h_k is estimated for data point x_k, the contribution of the distance $(x_i - x_k)^2$ to each data point x_i is normalised by the likelihood of x_i.

The effect on the bandwidth h_k of data points that fall within dense regions is thus reduced, while the effect of data points that lie in lower density regions is increased. This may thus be regarded as a form of bandwidth regularisation. To illustrate the regularisation effect, consider two data points, x_1 and x_2, that lie very close to each other. Both x_1 and x_2 will have relatively high density values for $p_{H(-1)}(x_1)$ and

$p_{H(-2)}(x_2)$, since they are close in feature space and will increase each other's density. If the bandwidth,h_1, is estimated for the kernel centred on x_1, the distance $(x_1 - x_2)^2$ will be very small, but since the value of $p_{H(-1)}(x_2)$ will be large, the contribution of $(x_1 - x_2)^2$ to h_1 will be reduced, thus preventing h_1 from becoming too small.

4.5.2.2 MULTIVARIATE GAUSSIAN KERNEL (FULL COVARIANCE MATRIX)

Similar to the univariate case, the multivariate Gaussian kernel function with a full covariance matrix , $\mathbf{H}_i$, can be expressed as the standard multivariate normal density function

$$K\left(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}_i\right) = \frac{1}{(2\pi)^{D/2}} e^{-\frac{1}{2}(\mathbf{x}_i-\mathbf{x}_j)^T \mathbf{H}_i^{-1}(\mathbf{x}_i-\mathbf{x}_j)}. \tag{4.59}$$

The multivariate Gaussian density function with full covariance matrix, $\mathbf{H}_i$, can be expressed as

$$K_{\mathbf{H}_i}\left(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}_i\right) = \frac{1}{(2\pi)^{D/2}\,|\mathbf{H}_i|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}_i-\mathbf{x}_j)^T \mathbf{H}_i^{-1}(\mathbf{x}_i-\mathbf{x}_j)} \tag{4.60}$$

and its partial derivative with respect to $\mathbf{H}_i$ as

$$\frac{\partial}{\partial \mathbf{H}_i} K_{\mathbf{H}_i}\left(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}_i\right) = \frac{1}{2}\mathbf{H}_i^{-2}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T K_{\mathbf{H}_i}\left(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}_i\right) - \frac{1}{2}\mathbf{H}_i^{-1} K_{\mathbf{H}_i}\left(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}_i\right) \tag{4.61}$$

We now obtain the partial derivative of the LOUT estimate in Eq. 4.45 by substituting the partial derivative of the kernel function in Eq. 4.61

$$\begin{aligned}\frac{\partial}{\partial \mathbf{H}_i}\left(p_{\mathbf{H}_i(-i)}(\mathbf{x}_i)\right) = \frac{1}{N-1}\left[\sum_{j\neq i} \frac{1}{2}\mathbf{H}_i^{-2}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T K_{\mathbf{H}_i}\left(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}_i\right)\right] \\ -\frac{1}{N-1}\left[\sum_{j\neq i} \frac{1}{2}\mathbf{H}_i^{-1} K_{\mathbf{H}_i}\left(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}_i\right)\right]\end{aligned} \tag{4.62}$$

We then substitute Eq. 4.62 into Eq. 4.49 to obtain the partial derivative of the MLL objective function, and set the derivative to zero to solve for $\mathbf{H}_i$

$$\mathbf{H}_i = \frac{\sum_{j\neq i}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T K_{\mathbf{H}_i}\left(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}_i\right)}{\sum_{j\neq i} K_{\mathbf{H}_i}\left(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}_i\right)}, \tag{4.63}$$

where $\mathbf{H}_i$ is the MLL full covariance bandwidth matrix estimate for the kernel centred on data point $\mathbf{x}_i$.

The time complexity of evaluating the multivariate Gaussian kernel, $K\left(x_i - x_j|H_i\right)$, with full covariance matrix, H_i, is of order $O(D^3)$ if Gauss-Jordan elimination is used to calculate the inverse of H_i and LU decomposition is used to calculate the determinant of H_i. Therefore, the time complexity of evaluating Eq. 4.63 will be of order $O(ND^3)$ owing to the $N-1$ evaluations of the kernel function in the numerator and denominator.

Since H_i must be calculated for each of the N training data points, the time complexity of the MLL full covariance estimator is of order $O(N^2D^3)$.

The parameters to estimate are the $D(D+1)/2$ parameters for each of the N symmetric full covariance matrices; therefore the number of parameters is $ND(D+1)/2$.

4.5.2.3 MULTIVARIATE GAUSSIAN KERNEL (DIAGONAL COVARIANCE MATRIX)

The partial derivative of the multivariate Gaussian expression in Eq. 4.32 with respect to the bandwidth matrix $\mathbf{H}_i$ in dimension d can be obtained by simply substituting h_{jd} in Eq. 4.33 with h_{id} to give

$$\frac{\partial}{\partial h_{id}}\left[K_{\mathbf{H}_i}\left(\mathbf{x}_i-\mathbf{x}_j|\mathbf{H}_i\right)\right]=\frac{1}{h_{id}^2}\left[\frac{K_{h_{id}}\left(\frac{x_{id}-x_{jd}}{h_{id}}\right)(x_{id}-x_{jd})^2}{h_{id}^2}-K_{h_{id}}\left(\frac{x_{id}-x_{jd}}{h_{id}}\right)\right]\prod_{p\neq d}K_{h_{ip}}\left(\frac{x_{ip}-x_{jp}}{h_{ip}}\right) \tag{4.64}$$

If we substitute this derivative into Eq. 4.49, then the partial derivative of the MLL objective function with respect to the diagonal bandwidth matrix $\mathbf{H}_i$ in dimension $d$is

$$\frac{d}{dh_{id}}\left(l_{MLL}(\mathbf{X})\right)=\frac{1}{(N-1)h_{id}^2}\sum_{j\neq i}\left[\frac{K_{h_{id}}\left(\frac{x_{id}-x_{jd}}{h_{id}}\right)(x_{id}-x_{jd})^2}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)h_{id}^2}-\frac{K_{h_{id}}\left(\frac{x_{id}-x_{jd}}{h_{id}}\right)}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)}\right]\prod_{p\neq d}K_{h_{ip}}\left(\frac{x_{ip}-x_{jp}}{h_{ip}}\right) \tag{4.65}$$

where h_{id} is the bandwidth of the univariate Gaussian kernel $K_{h_{id}}(\cdot)$ centred on data point $\mathbf{x}_i$ in dimension d.

We then set the derivative in Eq. 4.65 to zero and solve for h_{id}^2

$$h_{id}^2=\frac{\sum_{j\neq i}K_{h_{id}}\left(\frac{x_{id}-x_{jd}}{h_{id}}\right)(x_{id}-x_{jd})^2\prod_{p\neq d}K_{h_{ip}}\left(\frac{x_{ip}-x_{jp}}{h_{ip}}\right)}{\sum_{j\neq i}K_{h_{id}}\left(\frac{x_{id}-x_{jd}}{h_{id}}\right)} \tag{4.66}$$

Note that the term $p_{\mathbf{H}_i(-i)}(\mathbf{x}_i)$ in Eq. 4.65 may be moved outside of the summation since it is not a function of j, and therefore this term is cancelled out when Eq. 4.65 is set to zero.

If we substitute the definition of the multivariate Gaussian density function in Eq. 4.32 into Eq. 4.66, we obtain the solution of the d^{th} diagonal component of the MLL diagonal bandwidth matrix centred on data point $\mathbf{x}_i$

$$\mathbf{H}_{i(d,d)}=\frac{\sum_{j\neq i}K_{\mathbf{H}_i}\left(\mathbf{x}_i-\mathbf{x}_j|\mathbf{H}_i\right)(x_{id}-x_{jd})^2}{\sum_{j\neq i}K_{\mathbf{H}_i}\left(\mathbf{x}_i-\mathbf{x}_j|\mathbf{H}_i\right)} \tag{4.67}$$

The time complexity of evaluating the multivariate Gaussian kernel, $K_{H_i}(x_i-x_j|H_i)$, with diagonal covariance matrix, H_i, is of order $O(D)$ since the inverse of the covariance matrix is simply the inverse of each of the diagonal

elements and the determinant is simply the product of the diagonal elements. Eq. 4.67 requires the $N-1$ calculations of the kernel function, $K_{H_i}(x_i - x_j|H_i)$, which yields a time complexity of order $O(ND)$ for Eq. 4.67. Since H_i must be calculated for the N training data points, the time complexity of the diagonal MLL estimator is of order $O(N^2D)$.

The parameters to estimate are the D parameters for each of the N diagonal covariance matrices; therefore the number of parameters to estimate is ND. The parameters thus scale linearly with the number of training data points and the dimensionality of the dataset.

4.5.3 ML-LOO KERNEL BANDWIDTH ESTIMATION

We have thus far derived MLL and MLE bandwidth estimators from the ML LOUT framework and in this section we will show that the ML-LOO bandwidth estimation algorithm that was recently introduced by Leiva-Murillo and Artés-Rodríguez [18], can also be derived from the same ML framework. We perform the derivations of these algorithms since the full proofs of these algorithms were not published and this will allow us to compare the ML-LOO estimators to the MLE and MLL algorithms in a common theoretical framework.

Leiva-Murillo and Artés-Rodríguez derived the ML-LOO estimator for the full covariance and spherical covariance Gaussian kernels. The main restriction placed on the ML-LOO estimator is that an identical bandwidth is estimated for all kernels, as opposed to the MLL and MLE estimators that estimate a unique bandwidth matrix for each kernel.

We will compare both variants of the ML-LOO estimator to the MLE and MLL estimators in our simulation studies to investigate the effect of this simplification.

4.5.3.1 SPHERICAL ML-LOO ESTIMATOR

The LOUT KDE for a multivariate spherical kernel can be defined as

$$p_{h(-i)}(\mathbf{x}_i) = \frac{1}{N-1}\sum_{j\neq i}^{N} K(\mathbf{x}_i - \mathbf{x}_j|h) \tag{4.68}$$

where h is the spherical bandwidth that is identical for all kernels in all dimensions. The spherical ML-LOO objective function can thus be written as

$$\begin{aligned} l_{ML-LOO(s)}(X) &= \textstyle\sum_{i=1}^{N} \log\left[p_{h(-i)}(\mathbf{x}_i)\right] \\ &= \textstyle\sum_{i=1}^{N} \log\left[\frac{1}{N-1}\sum_{j\neq i}^{N} K(\mathbf{x}_i - \mathbf{x}_j|h)\right] \end{aligned} \tag{4.69}$$

and the derivative of this objective function with respect to the spherical bandwidth h can be expressed as

$$\begin{aligned}\frac{d}{dh}\left(l_{ML-LOO(s)}(X)\right) &= \sum_{i=1}^{N}\frac{d}{dh}\left(\log\left[p_{h(-i)}(\mathbf{x}_i)\right]\right)\\ &= \sum_{i=1}^{N}\frac{\frac{d}{dh}\left(p_{h(-i)}(\mathbf{x}_i)\right)}{p_{h(-i)}(\mathbf{x}_i)}\\ &= \sum_{i=1}^{N}\frac{\frac{1}{N-1}\sum_{j\neq i}\frac{d}{dh}K(\mathbf{x}_i-\mathbf{x}_j|h)}{p_{h(-i)}(\mathbf{x}_i)}\end{aligned} \tag{4.70}$$

The optimal spherical ML-LOO bandwidth can now be obtained by setting the derivative of the objective function in Eq. 4.70 to zero and solving for h.

The multivariate Gaussian density function with spherical covariance matrix can be expressed as

$$K_h\left(\mathbf{x}_i-\mathbf{x}_j|h\right) = \frac{1}{h^D(2\pi)^{D/2}}e^{-\frac{1}{2}\frac{\left\|\mathbf{x}_i-\mathbf{x}_j\right\|^2}{h^2}} \tag{4.71}$$

where h is the standard deviation or bandwidth used in all dimensions and $\|\mathbf{x}_i-\mathbf{x}_j\|^2$is the squared D-dimensional Euclidean distance between data points $\mathbf{x}_i$ and $\mathbf{x}_j$.

The derivative of the spherical multivariate Gaussian expression in Eq. 4.5.1.31 with respect to the bandwidth h, can be derived by means of the product rule to obtain

$$\frac{d}{dh}\left[K_h\left(\mathbf{x}_i-\mathbf{x}_j|h\right)\right] = K_h\left(\mathbf{x}_i-\mathbf{x}_j|h\right)h^{-3}\left\|\mathbf{x}_i-\mathbf{x}_j\right\|^2 - DK_h\left(\mathbf{x}_i-\mathbf{x}_j|h\right)h^{-1} \tag{4.72}$$

If we substitute this derivative into the spherical ML-LOO objective function in Eq. 4.70, set the objective function to zero and solve for h we obtain the following expression for the spherical ML-LOO bandwidth estimate

$$h^2 = \frac{1}{D}\frac{\sum_{i=1}^{N}\frac{1}{p_{h(-i)}(\mathbf{x}_i)}\sum_{j\neq i}K\left(\mathbf{x}_i-\mathbf{x}_j|h\right)\left\|\mathbf{x}_i-\mathbf{x}_j\right\|^2}{\sum_{i=1}^{N}\frac{1}{p_{h(-i)}(\mathbf{x}_i)}\sum_{j\neq i}K\left(\mathbf{x}_i-\mathbf{x}_j|h\right)} \tag{4.73}$$

Leiva-Murillo and Artés-Rodríguez further simplify this expression in [18] by substituting

$$\sum_{j\neq i}K\left(\mathbf{x}_i-\mathbf{x}_j|h\right) = (N-1)p_{h(-i)}(\mathbf{x}_i) \tag{4.74}$$

to obtain the following expression for the spherical ML-LOO bandwidth estimate

$$h^2 = \frac{1}{D(N-1)N}\sum_{i=1}^{N}\frac{1}{p_{h(-i)}(\mathbf{x}_i)}\sum_{j\neq i}K\left(\mathbf{x}_i-\mathbf{x}_j|h\right)\left\|\mathbf{x}_i-\mathbf{x}_j\right\|^2 \tag{4.75}$$

The time complexity of evaluating the multivariate Gaussian kernel, $K\left(x_i-x_j|h\right)$, with spherical covariance, h, is of order $O(D)$ owing to the calculation of the D-dimensional squared Euclidean distance in Eq. 4.71. The calculation of the leave-one-out likelihood, $p_{h(-i)}(x_i)$, requires $N-1$ evaluations of the spherical Gaussian

kernel and thus has a time complexity of order $O(ND)$. The calculation of the term $\sum_{j\neq i} K\left(x_i - x_j|h\right)\left\|x_i - x_j\right\|^2$ in Eq. 4.73 also requires $N-1$ evaluations of the spherical kernel and the D-dimensional squared Euclidean distance. This term as well as the leave-one-out likelihood term requires N evaluations in Eq. 4.73 and the simplified version in Eq. 4.75, which yields a time complexity of order $O(N^2D)$ for the spherical ML-LOO estimator.

The spherical bandwidth, that is identical for all kernels, is the only parameter that needs to be estimated. The number of parameters to estimate thus does not scale with the number of training samples or the number of dimensions.

4.5.3.2 FULL COVARIANCE ML-LOO ESTIMATOR

The LOUT KDE for a multivariate full covariance kernel can be defined as

$$p_{\mathbf{H}(-i)}(\mathbf{x}_i) = \frac{1}{N-1}\sum_{j\neq i}^{N} K\left(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}\right) \tag{4.76}$$

where $\mathbf{H}$ is the full covariance bandwidth matrix that is identical for all kernels. The full covariance ML-LOO objective function can thus be written as

$$\begin{aligned} l_{ML-LOO(f)}(\mathbf{X}) &= \textstyle\sum_{i=1}^{N} \log\left[p_{\mathbf{H}(-i)}(\mathbf{x}_i)\right] \\ &= \textstyle\sum_{i=1}^{N} \log\left[\frac{1}{N-1}\sum_{j\neq i}^{N} K\left(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}\right)\right] \end{aligned} \tag{4.77}$$

and the partial derivative of this objective function with respect to the full covariance bandwidth matrix $\mathbf{H}$ can be expressed as

$$\begin{aligned} \tfrac{\partial}{\partial \mathbf{H}}\left(l_{ML-LOO(f)}(\mathbf{X})\right) &= \textstyle\sum_{i=1}^{N} \tfrac{\partial}{\partial \mathbf{H}}\left(\log\left[p_{\mathbf{H}(-i)}(\mathbf{x}_i)\right]\right) \\ &= \textstyle\sum_{i=1}^{N} \frac{\frac{\partial}{\partial \mathbf{H}}\left(p_{\mathbf{H}(-i)}(\mathbf{x}_i)\right)}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)} \\ &= \textstyle\sum_{i=1}^{N} \frac{\frac{1}{N-1}\sum_{j\neq i}\frac{\partial}{\partial \mathbf{H}} K(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H})}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)} \end{aligned} \tag{4.78}$$

The optimal full covariance ML-LOO bandwidth can now be obtained by setting the derivative of the objective function in Eq. 4.78 to zero and solving for $\mathbf{H}$.

The multivariate Gaussian density function with full covariance matrix is given in Eq. 4.17 and its partial derivative with respect to the full covariance matrix $\mathbf{H}$ in Eq. 4.28. If we substitute the partial derivative in Eq. 4.28 into the numerator of the derivative of the full covariance ML-LOO objective function in Eq. 4.78, we obtain

$$\begin{aligned} \tfrac{\partial}{\partial \mathbf{H}}\left(p_{\mathbf{H}(-i)}(\mathbf{x}_i)\right) = \tfrac{1}{N-1}&\left[\textstyle\sum_{j\neq i}\frac{1}{2}\mathbf{H}^{-2}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T K_{\mathbf{H}}\left(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}\right)\right] \\ -\tfrac{1}{N-1}&\left[\textstyle\sum_{j\neq i}\frac{1}{2}\mathbf{H}^{-1} K_{\mathbf{H}}\left(\mathbf{x}_i - \mathbf{x}_j|\mathbf{H}\right)\right] \end{aligned} \tag{4.79}$$

If we substitute the partial derivative in Eq. 4.79 into the derivative of the full covariance ML-LOO objective function in Eq. 4.78 and set the derivative to zero, we obtain

$$\mathbf{H} = \frac{\sum_{i=1}^{N} \frac{1}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)} \sum_{j \neq i} (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T K_{\mathbf{H}} (\mathbf{x}_i - \mathbf{x}_j | \mathbf{H})}{\sum_{i=1}^{N} \frac{1}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)} \sum_{j \neq i} K_{\mathbf{H}} (\mathbf{x}_i - \mathbf{x}_j | \mathbf{H})}. \tag{4.80}$$

Leiva-Murillo and Artés-Rodríguez also simplify this expression by substituting Eq. 4.74 to obtain the following expression for the full covariance ML-LOO bandwidth estimate

$$\mathbf{H} = \frac{1}{N(N-1)} \sum_{i=1}^{N} \frac{1}{p_{\mathbf{H}(-i)}(\mathbf{x}_i)} \sum_{j \neq i} (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T K_{\mathbf{H}} (\mathbf{x}_i - \mathbf{x}_j | \mathbf{H}) \,. \tag{4.81}$$

Eq. 4.73 and Eq. 4.80 show that both variants of the ML-LOO estimator estimate an identical kernel bandwidth for all kernels. We will compare these estimators to the MLL and MLE estimators in our simulation study in Chapter 6, to determine whether these estimators perform competitively. The time complexity of evaluating the multivariate Gaussian kernel, $K_H (x_i - x_j | H)$, with full covariance matrix, H, is of order $O(D^3)$ if Gauss-Jordan elimination is used to calculate the inverse of H and LU decomposition is used to calculate the determinant of H. Therefore, the time complexity of evaluating Eq. 4.79 will be of order $O(N^2 D^3)$, owing to the $N-1$ evaluations of the kernel function in the calculation of $p_{H(-i)}(x_i)$, and the N evaluations of $p_{H(-i)}(x_i)$ in Eq. 4.79.

The full covariance bandwidth matrix, H, is identical for all kernels, and $D(D+1)/2$ parameters are estimated since the covariance matrix is symmetric. The estimated number of parameters thus does not scale with the number of training points and scales quadratically with dimensionality.

4.5.4 SUMMARY OF ML KDES COMPUTATIONAL COMPLEXITIES AND PARAMETERS TO BE ESTIMATED

Table 4.1 summarises the computational complexities and the number of parameters to be estimated as a function of N and D for all variants of the MLE, MLL and ML-LOO estimators that were derived in Sections 4.5.1 – 4.5.3

Table 4.1: Summary of computational complexities and parameters to be estimated for ML KDEs

Algorithm	Bandwidth Type	Computational Complexity	Parameters Estimated
MLE	Univariate	$O(N^3)$	N
MLE	Full covariance	$O(N^3D^3)$	$ND(D+1)/2$
MLE	Diagonal covariance	$O(N^3D)$	ND
MLE	Global diagonal covariance	$O(N^2D)$	D
MLL	Univariate	$O(N^2)$	N
MLL	Full covariance	$O(N^2D^3)$	$ND(D+1)/2$
MLL	Diagonal covariance	$O(N^2D)$	ND
ML-LOO	Global spherical covariance	$O(N^2D)$	1
ML-LOO	Global full covariance	$O(N^2D^3)$	$D(D+1)/2$

4.6 KERNEL BANDWIDTH INITIALISATION

We have shown that all ML bandwidth estimation procedures derived in Section 4.5 required a form of bandwidth initialisation to initialise the optimisation procedure for bandwidth estimation.

Barnard [17] and Leiva-Murillo and Artés-Rodríguez [18] made interesting observations regarding the range in which the optimal ML LOUT bandwidths should lie, by evaluating the zero-variance and infinite-variance limits of the MLL and ML-LOO bandwidth estimation equations. Leiva-Murillo and Artés-Rodríguez rewrote the bandwidth estimation equation for the spherical case in Eq. 4.75 as

$$g\left(h^2\right) = \frac{1}{ND}\sum_i \sum_{j\neq i} \frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{1 + \sum_{k\neq i,j} e^{\left(\frac{\left\|\mathbf{x}_i-\mathbf{x}_j\right\|^2 - \left\|\mathbf{x}_i-\mathbf{x}_k\right\|^2}{2h^2}\right)}} \tag{4.82}$$

and calculated the limit of $g\left(h^2\right)$ at zero variance

$$\lim_{h^2\to 0} g\left(h^2\right) = \frac{d_{NN}^2}{D} \tag{4.83}$$

where d_{NN}^2 is the squared nearest-neighbour D-dimensional Euclidean distance between any two data points in the dataset. They also calculated the limit of $g\left(h^2\right)$ at infinite variance as

$$\lim_{h^2\to \infty} g\left(h^2\right) = \frac{d^2}{D} \tag{4.84}$$

where d^2 is the average squared D-dimensional Euclidean-distance between all data points in the dataset.

To prove that the estimated kernel bandwidth, h^2, will always be in the range $[d_{NN}^2/D, d^2/D]$, Leiva-Murillo and Artés-Rodríguez also showed that $g(h^2)$ is monotonic in the interval, thus showing that the smallest possible estimated bandwidth value is d_{NN}^2/D and the largest possible estimated bandwidth is d^2/D for the spherical Ml-LOO bandwidth estimator.

Based on these theoretical bounds, we propose a rule-of-thumb bandwidth initialisation method, which we will call the MLE rule-of-thumb estimator. The MLE rule-of-thumb estimator simply estimates the initial bandwidth of a kernel between the theoretical lower and upper bounds.

More formally, the bandwidth $h_{i(0)}$ is estimated as

$$h_{i(0)} = \sqrt{\frac{d_{NN}^2 + d^2}{2}} \tag{4.85}$$

where d_{NN}^2 is the squared Euclidean distance from x_i to its nearest-neighbour, and d^2 is the squared average Euclidean distance from x_i to all other data points. For the multivariate case, the nearest neighbour to data point $\mathbf{x}_i$ is be selected based on a D-dimensional Euclidean distance, and the initial bandwidths are calculated independently per dimension. Thus Eq. 4.85 can be used for each dimension independently.

We will compare the MLE rule-of-thumb estimator to other rule-of-thumb estimators in our simulation study in the next chapter.

4.7 CONCLUSION

In this chapter we have shown how different formulations of LOUT kernel density estimation can be used to define various ML objective functions. We also showed how these objective functions can be used to derive ML bandwidth estimation algorithms. In particular, we proved the existing MLL and ML-LOO bandwidth estimators, and we derived two novel MLE bandwidth estimators from first principles. We also proposed a rule-of-thumb bandwidth initialisation procedure based on the theoretical bounds of ML bandwidth estimators.

In the next chapter we will perform a comparative simulation study of conventional kernel bandwidth estimators. We will also include the MLE rule-of-thumb estimator in the comparative study to gain a better understanding of its performance in practice. In Chapter 6 we will perform a comparative simulation study of all bandwidth estimators that were derived in this chapter.

Chapter Five

Comparison of Conventional Kernel Bandwidth Estimators

Contents

5.1 Introduction 61

5.2 Methods And Data 61

5.2.1 Data sampling procedure 61

5.2.2 Cross-validation data 63

5.2.3 Real-world data 64

5.2.4 Data pre-processing 66

5.2.5 Methods 68

5.2.6 Performance evaluation 68

5.3 Experimental design 69

5.3.1 Experiment 1 69

5.3.2 Experiment 2 71

5.3.3 Experiment 3 71

5.4 Results 71

5.4.1 Experiment 1 71
5.4.2 Experiment 2 81
5.4.3 Experiment 3 85
5.5 Conclusion 92

5.1 INTRODUCTION

In this chapter we compare the performance of conventional KDE bandwidth estimators on numerous artificial and RW data sets. We simulate artificial data to investigate the performance of bandwidth estimators under various conditions. We structure our simulations to investigate the following data properties: dimensionality, number of samples, modality, scale, correlation and anisotropy. As discussed in Section 3.4 and [52, 26, 47], these properties are fundamental data properties that should be captured by estimators to describe data accurately. The advantage of using artificial data is that these properties can be manipulated in a controlled fashion and to various degrees. We can thus perform estimation on the sampled artificial datasets to probe the performance of the estimators as the data properties are varied.

We also compare the performance of the bandwidth estimators on RW datasets. We distinguish between two types of RW datasets, namely datasets with no independent test set, and datasets with separate train and test sets. We will refer to the RW datasets with no independent test sets as CV datasets, and to the datasets with separate train and test sets simply as RW datasets.

We describe the methodology for artificial data generation, the methods that will be compared and the data pre-processing in Section 5.2, and design our experiments for artificial, CV and RW data simulation studies in Section 5.3; we present the results of the simulation studies in Section 5.4 and conclude our findings in Section 5.5.

5.2 METHODS AND DATA

5.2.1 DATA SAMPLING PROCEDURE

In order to probe the performance of bandwidth estimators for varying data properties, we sample artificial datasets from the multivariate normal (MVN) distribution. The MVN distribution only requires the specification of a mean and covariance matrix, and we explain in this section how various data properties can be modelled by introducing linear transformations to generate the appropriate covariance matrices.

We initialise the sampling process by sampling data from a standard MVN distribution with 0 mean and the identity matrix as covariance (i.e. a standard deviation of 1 in all dimensions). We introduce scale into the data with a simple linear transformation

$$\mathbf{Y}_{diag} = \mathbf{A}_{diag}\mathbf{X} \tag{5.1}$$

where $\mathbf{X}$ is the data sampled from a standard MVN distribution and $\mathbf{A}$ is the scaling matrix such that the covariance of the transformed data $\mathbf{Y}_{diag}$ is equal to

$$\mathbf{\Sigma}_{diag} = \mathbf{A}_{diag}\mathbf{A}_{diag}^{T} \tag{5.2}$$

$\mathbf{A}_{diag}$ is a diagonal matrix, where the diagonal components are the standard deviations of the transformed features. Consequently, the resulting covariance matrix will also be diagonal. The linearly transformed data $\mathbf{Y}_{diag}$thus also has dimensionality D, and are scaled according to the diagonal covariance matrix, $\mathbf{\Sigma}_{diag}$.

We now introduce correlation into the data by first generating an orthogonal feature space with the Gram Schmidt orthogonalisation procedure. The Gram Schmidt procedure is required to ensure that the standard deviations of the features in the transformed space remain the same as the standard deviations in $\mathbf{\Sigma}_{diag}$.

We thus generate an orthogonal D-dimensional scale matrix $\mathbf{A}_{GS}$ from the Gram Schmidt procedure, and construct a covariance matrix with orthogonal dimensions as follows:

$$\mathbf{\Sigma}_{GS} = \mathbf{A}_{GS}\mathbf{A}_{GS}^{T}. \tag{5.3}$$

In order to introduce correlation into the transformed data $\mathbf{Y}_{diag}$, we perform another linear transformation. The linear transformation is obtained by performing an eigenvalue transformation on $\mathbf{\Sigma}_{GS}$. This linear transformation will introduce correlation into the data, but will ensure that the resulting covariance matrix of the transformed data is of full rank. If the eigenvectors of the eigenvalue transformation are denoted by $\mathbf{V}$, the transformation can be expressed as

$$\mathbf{Y}_{full} = \mathbf{Y}_{diag}\mathbf{V} + \mathbf{B}\mathbf{V} \tag{5.4}$$

where $\mathbf{Y}_{full}$ is the linearly transformed data with eigenvalues equal to the variance values in $\mathbf{\Sigma}_{diag}$, and $\mathbf{B}$ is the mean of the data in the original feature space $\mathbf{Y}_{diag}$, that is then transformed to the new feature space containing correlation,$\mathbf{Y}_{full}$.

To introduce various modes into the data, we simply sample data with the above procedure, and add different mean values ($\mathbf{B}$) to the data, in order to perform translation of the data. We then concatenate the data of the various modes into a single data matrix.

We then introduce anisotropy to the data by making data sampled from different modes vary in different directions; more than one mode is thus required to make data anisotropic. More formally, we multiply the values of the first feature in $\mathbf{Y}_{full}$ with -1, which reverses the direction of the first dimension, thus ensuring that first features of the two modes vary in different directions. If we have two modes in a dataset, the data for the first mode will be generated with

$$\mathbf{Y}_{M1} = \mathbf{Y}_{trans} + \mathbf{B}_{M1}\mathbf{V} \tag{5.5}$$

where V is the eigenvector transformation obtained from $\mathbf{\Sigma}_{GS}$, $\mathbf{B}_{M1}$ is the mean of mode 1 in the diagonal feature space and where

$$\mathbf{Y}_{trans} = \mathbf{Y}_{diag}\mathbf{V}_{M1}. \tag{5.6}$$

The data for the second mode arethen generated with

$$\mathbf{Y}_{M2} = [-Y_{trans_1},\ Y_{trans_2},\ \cdots, Y_{trans_D}] + \mathbf{B}_{M2}\mathbf{V} \tag{5.7}$$

where Y_{trans_i} is the data for the i^{th} feature in dataset $\mathbf{Y}_{trans}$ and $\mathbf{B}_{M2}$ is the mean of mode 2 in the diagonal feature space.

This will thus ensure that the data for $\mathbf{Y}_{M1}$ and $\mathbf{Y}_{M2}$ will vary in different directions in the first dimension.We note that anisotropy is only relevant for multi-modal data where the features are correlated, since the change indirection of the first dimension will not change the distribution of uncorrelated data.

5.2.2 CROSS-VALIDATION DATA

We select eight CV datasets (with no independent test sets) from the UCI Machine Learning Repository [42] for the purpose of simulation studies. These datasets are selected to cover a wide range of applications in pattern recognition, and to be representative of the samples sizes and dimensionalities of typical pattern recognition problems. We briefly describe each of these datasets in this section, and summarise the most important dataset properties in Table 5.1.

The *Old Faithful* dataset consists of 272 observations of eruptions of the Old Faithful geyser in the Yellowstone national park. The observations consist of two measures, namely the duration since the previous eruption and the duration of the observed eruption. No class labels are assigned to the observations.

The *Balance Scale* dataset consists of 625 observations of a balance scale that is tipped to the left, balanced or tipped to the right. The observations consist of four

features that measure the left weight, distance to the left, right weight and distance to the right. The classification of each observation is the positioning of the scale, thus left, right or balanced.

The *Iris* dataset consists of 150 observations of iris flowers. Each observation consists of four measures of an iris flower, and the classification of each observation is the type of iris species to which the flower belongs, namely Iris Setosa, Iris Versicolour or Iris Virginica.

The *Diabetes* dataset consists of records of 768 patient records obtained from the National Institute of Diabetes and Digestive and Kidney Diseases. Each observation consists of eight features that may be used to predict the onset of diabetes. The classification of each observation is a diagnostic of whether the patient shows signs of diabetes according to the World Health Organization criteria.

The *Heart (Statlog)* dataset consists of 270 patient records that each consists of 13 features that may be used to predict the presence of heart disease. The classification of each observation is the absence or presence of heart disease in a patient.

The *Vehicle* dataset consists of 946 observations of vehicle silhouettes. The observations consist of 18 features that describe the scale-independent features and heuristic measures of a vehicle silhouette. The classification of each observation is the type of vehicle model, namely a bus, Chevrolet van, Saab 9000 or Opel Mantra 400.

The *Waveform* dataset consists of 5 000 observations from waveforms generated from two or three base waveforms with added noise of unit variance. Each observation is sampled from one of three waveforms and is described by 21 features. The classification of each observation is the type of waveform.

The *Ionosphere* dataset consists of 351 radar observations of the ionosphere measured at Goosebay, Labrador. The observations consist of 17 pairs of complex values resulting from the radar returns. There are thus 34 features per observation, and the classification of each observation is whether radar returns show evidence of structure in the ionosphere (measured by collisions with free electrons) or whether signals mostly passed through the ionosphere.

We summarise the CV datasets according to class, dimensionality and total number of samples in Table 5.1. The total number of classes are denoted as “C”, the number of dimensions as “D” and the total number of samples as “N”.

5.2.3 REAL-WORLD DATA

We selected five RW datasets (with independent train and test sets)for the purpose of simulation studies. These datasets were also obtained from the UCI Machine Learning

Table 5.1: CV dataset summary

Dataset	**C**	**D**	**N**
Old Faithful	1	2	272
Balance Scale	3	4	625
Iris	3	4	150
Diabetes	2	8	768
Heart (Statlog)	2	13	270
Vehicle	4	18	946
Waveform	3	21	5000
Ionosphere	2	33	351

Repository and were selected to have relatively high dimensionality, since these high dimensionalities are typical in RW pattern recognition tasks. We briefly describe each of these datasets in this section, and summarise the most important dataset properties in Table 5.2.

The *Letter* dataset consists of 16 000 training and 4 000 test observations of the 26 capital letters in English. Observations for 20 different fonts of each letter were generated and random distortions were introduced. The observations consist of 16 numerical features that are derived from statistical moments and edge counts. The classification of each observation is the letter category.

The Statlog Image *Segmentation* dataset consists of 2 100 train and 210 test observations, each corresponding to a $3x3$ pixel region randomly selected from seven outdoor images. The observations consist of 18 features that describe various properties of the region, such as the position, contrast with neighbouring pixels, intensity and colour properties of the region. The classification of each observation is the class of segment to which the centre pixel of the $3x3$ region belongs, compared to the segmentation class of hand segmentations. The segmentation classes are brickface, sky, foliage, cement, window, path and grass.

The *Landsat* satellite dataset consists of 4435 train and 2 000 test observations. Digital satellite images of the same scene were taken at four different spectral bands, two in the visible region and two in the near-infrared region. Each observation in the dataset corresponds to measurements in the four spectral bands for a $3x3$ pixel region, which results in 36 features per observation. The classification of each observation is the class of land coverage of the centre pixel in the $3x3$ region. The land coverage classes are: red soil, cotton crop, grey soil, damp grey soil, soil with vegetation stubble and very damp grey soil.

The *Optdigits* handwritten digits dataset consists of 3 823 train and 1 797 test observations. Handwritten digits of 43 people were used, 30 people for training and 13 people for testing. The image of each handwritten digit was divided into an 8x8 grid, resulting in 64 features per observation, where each feature describes the pixel intensity of a block in the $8x8$ grid. The classification of each observation is the digit value, which ranges from 0 to 9.

The *Isolet* spoken letters dataset consists of 6 238 train and 1 559 test observations. Two utterances of each letter in the English alphabet were recorded for 150 speakers. The utterances of 120 speakers are used in the train set and those of 30 speakers are used in the test set. Three utterances have been omitted from the dataset owing to recording difficulties. The observations consist of 617 features that describe various properties of a speech utterance; these features include spectral coefficients, contour features, sonant features, pre-sonant features and post-sonant features. The classification of each observation is the letter category.

We summarise the most important dataset properties of the RW datasets in Table 5.2. We denote the total number of classes as "C", the dimensionality of the dataset as "D", the total number of training samples as "NCtr" and the total number of test samples as "NCte".

Table 5.2: RW dataset summary

Dataset	**C**	**D**	**NCtr**	**NCte**
Letter	26	16	16000	4000
Segmentation	7	18	2100	210
Landsat	6	36	4435	2000
Optdigits	10	64	3823	1797
Isolet	26	617	6238	1559

5.2.4 DATA PRE-PROCESSING

PCA is performed on RW and CV datasets as a pre-processing step prior to density estimation. PCA is used to reduce the dimensionality of datasets by performing a linear transformation that re-aligns the feature axes to the directions of most variation, and thus minimises the variance orthogonal to the projection. Features with the smallest eigenvalues (or variance in the transformed feature space) may thus be disregarded. PCA also ensures that the features of the transformed feature space are orthogonal, thus ensuring the features are decorrelated.

A more subtle implication of PCA is that it serves as a form of bandwidth regularisation for ML kernel bandwidth estimators, which is a crucial task for kernel density estimation in high-dimensional spaces. Because of the formulation of the ML objective function, an infinite likelihood score is obtained when two data points have corresponding values in any dimension. When, for example, a Gaussian kernel is fitted over a training data point, and a test point has the same value in any of the dimensions, the bandwidth that will maximise the likelihood score in the dimension with corresponding values will be 0. This phenomenon becomes more problematic as dimensionality increases, since the probability of having two observations with the same value in any dimension increases significantly, thus leading to more degenerate kernel bandwidths.

PCA reduces the probability of identical values in a dimension since the linear transformation of a new feature is the weighted sum of the values of all dimensions in the original feature space. Two data points thus need to satisfy the same linear equation(i.e. have identical values in all dimensions) to have the same value in a transformed dimension.

The dimensionality of all RW and CV datasets can be reduced with PCA in two ways. The first approach is to remove all transformed features that have eigenvalues smaller than 1% of the eigenvalue of the principal component. An alternative approach is to keep the features with largest eigenvalues such that their eigenvalues sum to 95% of the total of all eigenvalues. This approach attempts to capture 95% of the variance, but fails when, for example, all the features have approximately the same eigenvalues. In this case, features will be removed that do not necessarily have much smaller eigenvalues than the selected features. This scenario is prevented with the first approach that retains all features with eigenvalues larger than 1% of the principal component eigenvalue. We make use of this approach in our experiments, and denote this pre-processing step for dimensionality reduction as PCA1.

We employ a second dimensionality reduction approach in our experiments that selects only the five transformed features with the largest eigenvalues. This approach allows us to compare estimators in this relatively low dimensional space, since in some instances the first dimensionality reduction approach will still select more than 10 transformed features. We denote this pre-processing step for dimensionality reduction as PCA5.

We also consider two variants of PCA transformations for PCA5 and PCA1, namely class-specific PCA and global PCA. Class-specific PCA transforms each class in a dataset independently and thus provides a unique transformation per class that decorrelates the features for each class. This transformation has proved to be more effective[48]

in compressing features, than when all classes are transformed simultaneously. Global PCA transforms all classes simultaneously and uses the same transformation across all classes. We report on class-specific PCA results in the body of this chapter and attach the global PCA results in Appendix A for comparative purposes.

5.2.5 METHODS

We make use of the conventional Silverman rule-of-thumb, MSP, LSCV, ULCV, LLCV and the HSJM plug-in rule estimators as implemented in the KDE toolbox[53] in our simulation studies.

The LSCV, ULCV and LLCV estimators employ a golden section search to optimise the bandwidth for the respective objective functions, and the search is initialised with a bandwidth based on the average nearest neighbour distance. All implementations assume a Gaussian kernel, and the Silverman estimator assumes a Gaussian reference distribution.

The LSCV, ULCV and LLCV estimator scale the features independently to one standard deviation prior to bandwidth estimation, and the optimal bandwidth is re-scaled per dimension according to the initial standard deviations after estimation. We denote scaling with (s) at the end of the names of the scaled estimators. Since PCA is performed as a pre-processing step on all data, this is necessary to prevent the over-smoothing of dimensions with small eigenvalues and the under-smoothing of dimensions with large eigenvalues, when only a single bandwidth is estimated per dimension.

We also implement the ML Gauss estimator, which simply estimates the sample mean and covariance of each class. We make use of a full covariance matrix estimate for dimensionalities between 1 and 10, and make use of a diagonal covariance matrix for dimensionalities higher than 10. Finally, we implement the MLE rule-of-thumb estimator introduced in Chapter 4 and denote this estimator as "MLE(i)" for the remainder of this chapter, where "i" indicates that this estimator is regarded as a bandwidth initialisation procedure for the purpose of this study.

5.2.6 PERFORMANCE EVALUATION

We sample independent test sets for artificial data to compare the performance of the estimators on these datasets. For each distribution, we sample 10 training and 10 test sets. We will discuss the experimental design of artificial data in more detail in the next section. A density is estimated for each of the 10 training sets, and for each training set a corresponding test set is used to calculate the likelihood scores of the test data

points. The likelihood scores of each test set are used to calculate an entropy score per test set, and the average of the 10 entropy scores is used to report on the final performance of the estimators.

We make use of 10-fold CV to estimate the performance of the estimators on CV datasets with no independent test sets. A density is estimated for each of the 10 folds, and the left-out test fold is used to calculate the likelihood scores of the test data points. After all 10 test folds have been evaluated, the likelihood scores are combined to estimate the entropy of the estimators.

We estimate a density function on the training set of RW datasets with independent training and test sets. The likelihood scores of the test set are obtained and combined to calculate the entropy of the test set.

5.3 EXPERIMENTAL DESIGN

We perform simulation studies on artificial, CV and RW datasets. Artificial datasets are generated as described in section 5.2.1, since this allows us to probe the performance of the estimators, as data properties are varied in a controlled fashion. We also experiment with the CV datasets described in Section 5.2.2, and evaluate the performance of estimators as described in Section 5.2.6. We use these datasets to gain a better understanding of the performance of the estimators over a wide range of application and under a varying number of dimensions and data points. We also experiment with the RW datasets with an independent test set, as described in Section 5.2.3. These test sets have been carefully selected by the authors of the datasets to test the generalisation performance of algorithms, and therefore we rely on the performance of estimators on these datasets to indicate how well the estimators can be expected to perform on other RW density estimation tasks.

5.3.1 EXPERIMENT 1

We generate seven sets of artificial data to investigate the effects of the following data properties on the estimators listed in Section 5.2.4: dimensionality, number of samples, modality, scale, correlation and anisotropy. We summarise the permutations of data properties for the seven artificial sets of data in Table 5.3.

Artificial Set 1 allows us to establish baseline performances for the estimators on the standard MVN distribution; this distribution has no scale variations within or between features, no correlation between features and no anisotropy, since there is only one mode.

Table 5.3: Summary of artificial dataset properties

Artificial Set	**Modes**	**Scaled**	**Correlation**	**Anisotropic**
Artificial 1	1	No	No	No
Artificial 2	1	Yes	No	No
Artificial 3	1	Yes	Yes	No
Artificial 4	2	No	No	No
Artificial 5	2	Yes	No	No
Artificial 6	2	Yes	Yes	No
Artificial 7	2	Yes	Yes	Yes

Artificial Set 2 allows us to probe the effect of scale variations on the estimators, since the standard deviations of the features are increased from 1 to D.

Artificial Set 3 allows us to probe the effect of correlation on the estimators, since the data are sampled from an MVN distribution with correlation between features. Note that scaling of features is also employed, since correlation between features will have no effect on a spherical distribution.

Artificial Set 4 allows us to probe the effect of bimodality on the estimators. The data are sampled from two modes and there are no scale variations within or between features, no correlation between features and no anisotropy, since the distributions of the modes are spherical.

Artificial Set 5 allows us to probe the effect of scale variations within features. This is accomplished by increasing the standard deviations of features in the first mode from 1 to D, and decreasing the standard deviations of features in the second mode from D to 1. Scale changes within features are thus simulated, since the sampled data will, for example, contain values sampled from the first mode with standard deviation 1, and the second mode with standard deviation D.

Artificial Set 6 consists of data sampled from two modes that have scale variations similar to Artificial Set 5 and correlation between features. The covariance matrices of the two modes, are, however, identical, thus variations will be in the same direction, which results in no anisotropy.

Artificial Set 7 consists of data sampled from two modes that have scale variations and correlation between features. The covariance matrices of the two modes are, however, not identical as in the case of Artificial Set 6 and are constructed such that the two modes vary in opposing directions in the first dimension, thus simulating a manifold with curvature.

For each of the artificial sets the dimensionalities of the distributions are varied

from 1 to 10 and 10 to 100 (in intervals of 10). We sample 10 training and 10 test sets per distribution for the purpose of our experiments, which results in 180 datasets per artificial set and 1 080 artificial datasets in total. The training sets contain 500 samples and the test sets 100 samples per dataset.

5.3.2 EXPERIMENT 2

We compare the performance of the estimators listed in Section 5.2.4 on the CV datasets described in Section 5.2.2, and we evaluate the performance of the estimators on the CV datasets as described in Section 5.2.6.We perform four variants of PCA pre-processing, namely class-specific PCA5, global PCA5, class-specific PCA1 and global PCA1, as described in Section 5.2.4.

5.3.3 EXPERIMENT 3

We compare the performance of the estimators listed in Section 5.2.4 on the RW datasets described in Section 5.2.3, and we evaluate the performance of the independent test set as described in Section 5.2.6.We again perform four variants of PCA pre-processing, namely class-specific PCA5, global PCA5, class-specific PCA1 and global PCA1, as described in Section 5.2.4.

5.4 RESULTS

In this section we present the results of the experiments, as discussed in Sections 5.3.1 – 5.3.3. We denote the MLE rule-of-thumb initialisation method as MLE(i), the scaled uniform LCV estimator as ULCV(s), the scaled local LCV estimator as LLCV(s), the scaled LSCV estimator as LSCV(s), the Silverman rule-of-thumb estimator as Silverman, the MSP estimator as MSP, the HSJM plug-in estimator as HSJM and the ML Gaussian estimator as ML Gauss.

5.4.1 EXPERIMENT 1

Figures 5.1-5.12 show the comparative entropy results of the conventional estimators (listed in Section 5.2.5) on Artificial Sets 1-7 for dimensionalities ranging from 1 to 10 and 10 to 100.

Artificial Set 1 contains data sets sampled from the standard MVN distribution. Figure 5.1 shows that the ML Gauss estimator performs best over all the dimensions, with the exception of the one-dimensional case where LLCV performs best. This is

not surprising, since the datasets are samples from the standard Gaussian distribution. The LSCV estimator performs significantly worse than the other estimators in the one-dimensional case; there might be different reasons for this performance, namely that the LSCV suffers from local minima and consequently the performance is very sensitive to the initial bandwidth selected for the optimisation procedure and that the LSCV estimator suffers from sample variability, which leads to highly variable bandwidth results for different samples sets from the same distribution. The HSJM estimator performs significantly worse than the other estimators for dimensionalities 3-7. This is surprising, since the HSJM plug-in uses a Gaussian distribution as reference distribution in the derivation of the plug-in rule, and the Silverman and MSP estimators (which are also derived from the AMISE) perform well. The MLE(i) estimator performs well in all dimensions, and the relative performance of this estimator seems to improve as dimensionality increases. This might be attributed to the fact that the estimator generally over-smooths bandwidths, and as dimensionality increases over-smoothing of bandwidths generally becomes more acceptable, since the volume of the feature space increases exponentially and larger bandwidths tend to lead to better generalisation performance. The MSP estimator performs relatively better than the Silverman estimator, and the relative difference in performance also increases with dimensionality. Since the MSP estimator over-smooths bandwidths, this confirms our intuition that larger bandwidths become more appropriate as dimensionality increases.

Figure 5.2 shows that LLCV and LSCV perform significantly worse than the other estimators on the high-dimensional standard MVN data. We also observe that the performance of the ML Gauss remains optimal for these high dimensions (even though a diagonal covariance matrix has been used for these dimensions). We also observe that the MSP and MLE(i) estimators consistently outperform the Silverman estimator in these high dimensions, thus verifying that larger bandwidths are more appropriate for higher dimensions.

In *Artificial Set 2*, we introduce scale variations by increasing the standard deviations of the features systematically. Figure 5.3 shows that the performances of the estimators are very similar for the first five dimensions. The LSCV(s) and LLCV(s) estimators, however, show degradation in performance relative to the other estimators, as the dimensionality increases beyond 5. This implies that these two estimators do not perform well when features of different scales are introduced (compared to the other estimators), even though bandwidths are scaled according to the standard deviations of the features. We also observe that the entropy scores of all estimators are generally higher than their entropy scores on Artificial Set 1.

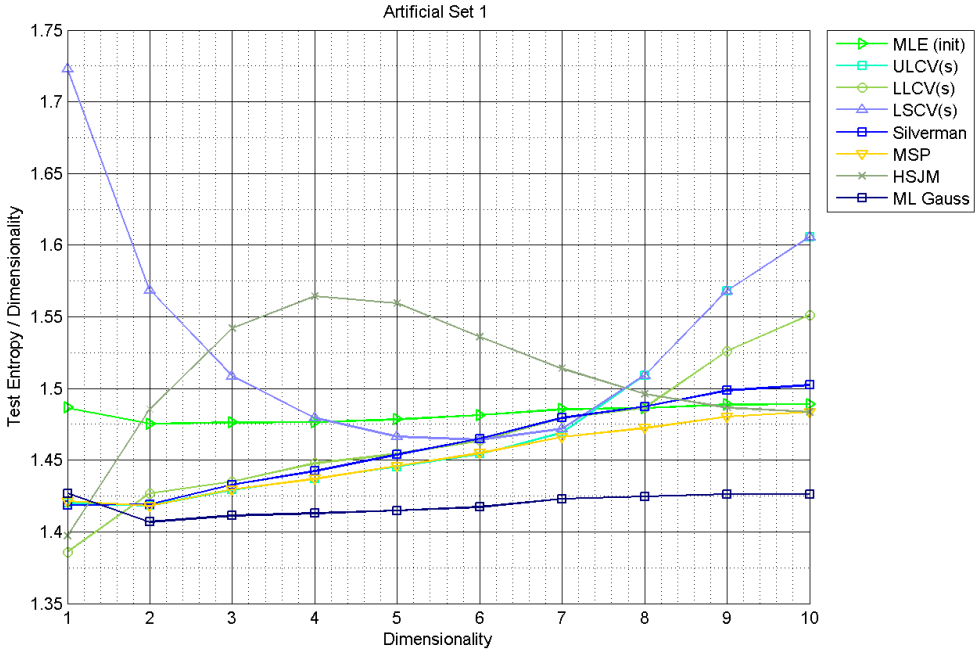

Figure 5.1: *Test entropy results for Artificial Set 1 (D 1-10)*

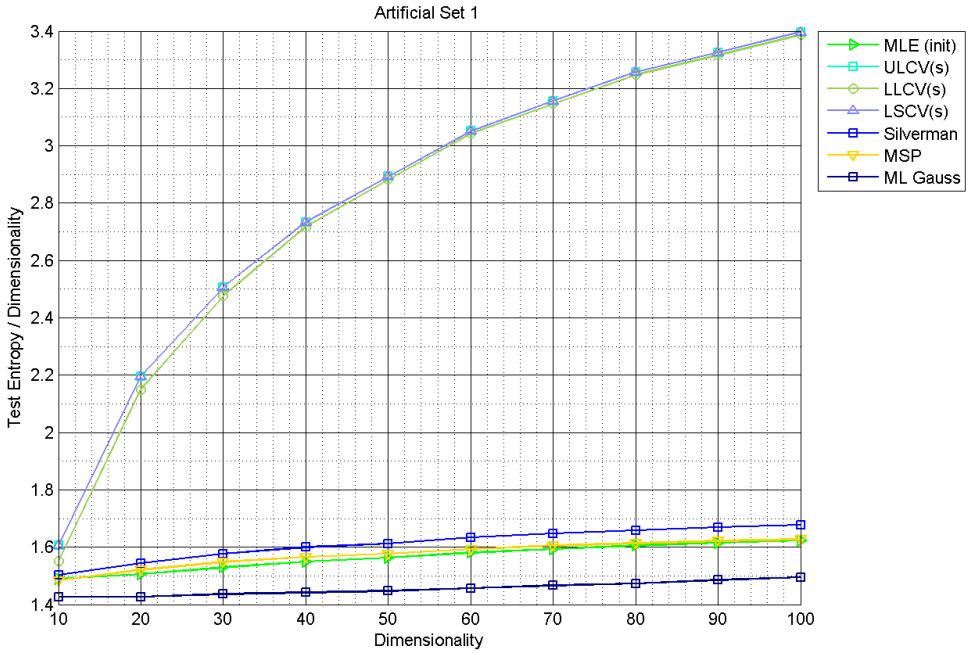

Figure 5.2: *Test entropy results for Artificial Set 1 (D 10-1000)*

In Figure 5.4 we observe similar performance behaviour to Artificial Set 1 in high dimensions. The LSCV and LLCV estimators perform significantly worse than the other estimators and the ML Gauss consistently outperforms the other estimators.

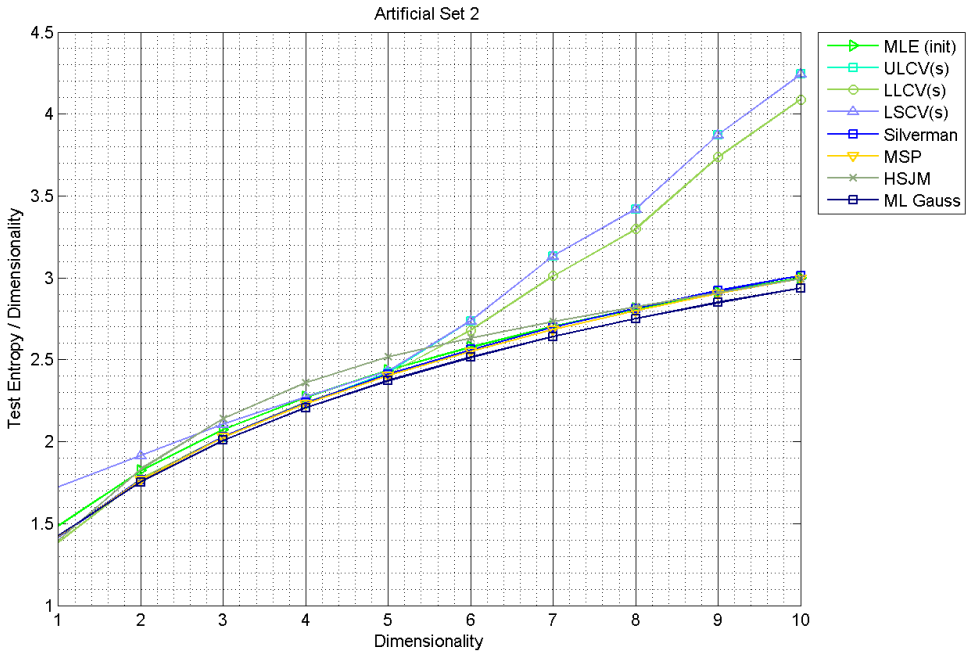

Figure 5.3: *Test entropy results for Artificial Set 2 (D 1-10)*

In *Artificial Set 3* we introduce correlation to the MVN distribution with features of different scales. Figure 5.5 shows that the ML Gauss performs best across all dimensions. This is to be expected, since the data sets are sampled from an MVN Gauss distribution and since the ML Gauss is capable of modelling features with different standard deviations and correlation by estimating a covariance matrix.

We observe in Figure 5.6 that the relative performance of the ML Gauss estimator increases even more as dimensionality increases, and that similar to Artificial Set 1 and Artificial Set 2, the LSCV and LLCV methods underperform significantly. The remaining estimators perform competitively.

In *Artificial Set 4*, we sample datasets from two standard MVN distributions to simulate bimodal data. Figure 5.7 shows that the relative performance of the MLE(i) estimator is significantly worse compared to the other estimators, except for the one-dimensional case where the LSCV(s) estimator performs worse. The bimodality of the datasets causes the MLE(i) to over-smooth the bandwidths, since the MLE(i) estimates the bandwidth for a kernel based on the average distances to all other data points. The distances from a data point (for which the kernel bandwidth is estimated) to data

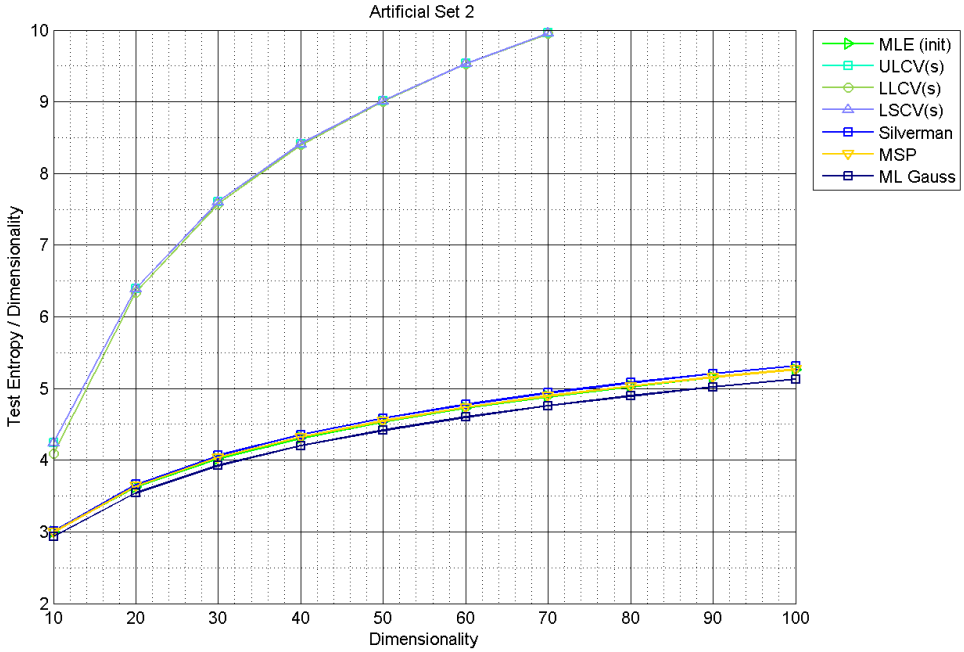

Figure 5.4: *Test entropy results for Artificial Set 2 (D 10-100)*

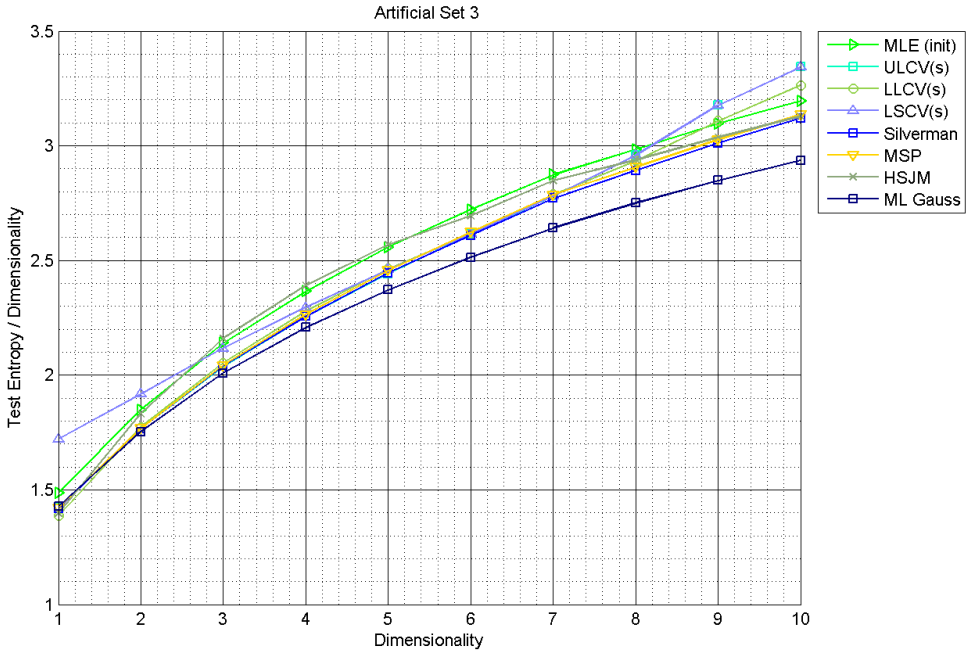

Figure 5.5: *Test entropy results for Artificial Set 3 (D 1-10)*

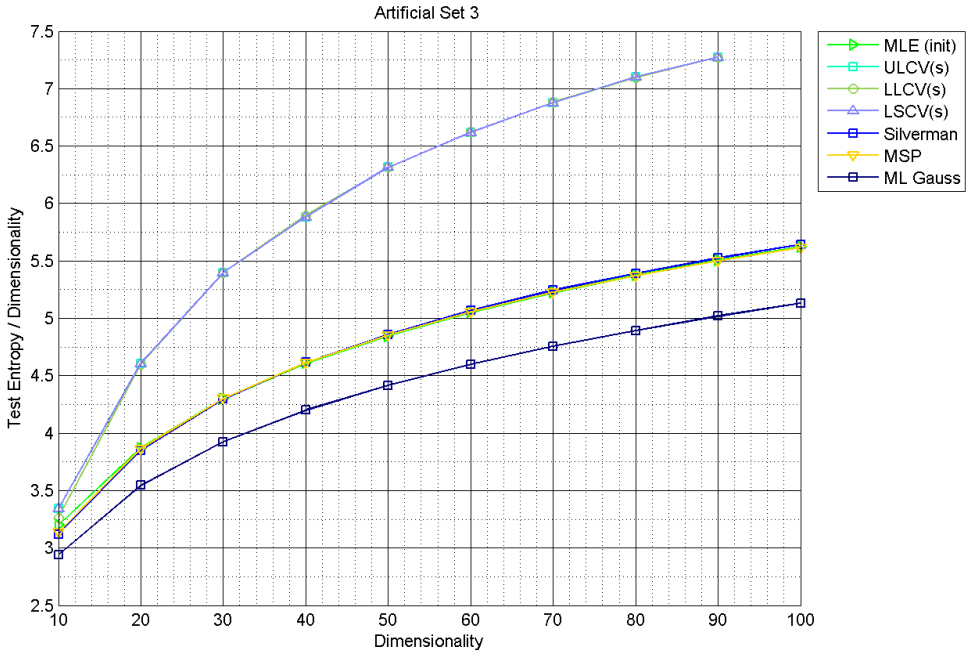

Figure 5.6: *Test entropy results for Artificial Set 3 (D 10-100)*

points belonging to a different mode will increase the average distance beyond what is optimal, thus leading to over-smoothed bandwidths. Ideally distances to data points should be weighed such that closer data points have more weight (as in the case of the MLL, MLE and ML-LOO estimators). Since the MLE(i) does not weigh the distances, the bandwidths are generally over-smoothed and even more so for multi-modal data.

We also observe that the performance of the ML Gauss estimator relative to the other estimators has decreased significantly compared to its relative performance on Artificial Sets 1-3. Since the ML Gauss assumes a unimodal distribution, this degradation in performance is to be expected. The performances of the remaining estimators are comparable, except for the LSCV(s) estimator in the one-dimensional case.

Figure 5.8 shows that the relative performance increase of the ML Gauss estimator increases as dimensionality increases, which is interesting, since the bimodality of the data does not fit the underlying assumptions of the ML Gauss estimator. It thus seems that as dimensionality increases, the effect of the bimodality on the ML Gauss estimator decreases. A surprising observation is that the Silverman estimator consistently outperforms the MSP estimator. This is surprising, since the MSP bandwidths are larger, and on the unimodal data sets, these larger bandwidths were advantageous in high dimensions. It thus seems that the Silverman estimator is more suitable for bimodal data than the MSP estimator, even though both estimators are derived from an

MVN reference distribution. The relative performance of the MLE(i) estimator seems to improve with dimensionality, which is consistent with the theory that bandwidth over-smoothing becomes more appropriate with an increase in dimensionality.

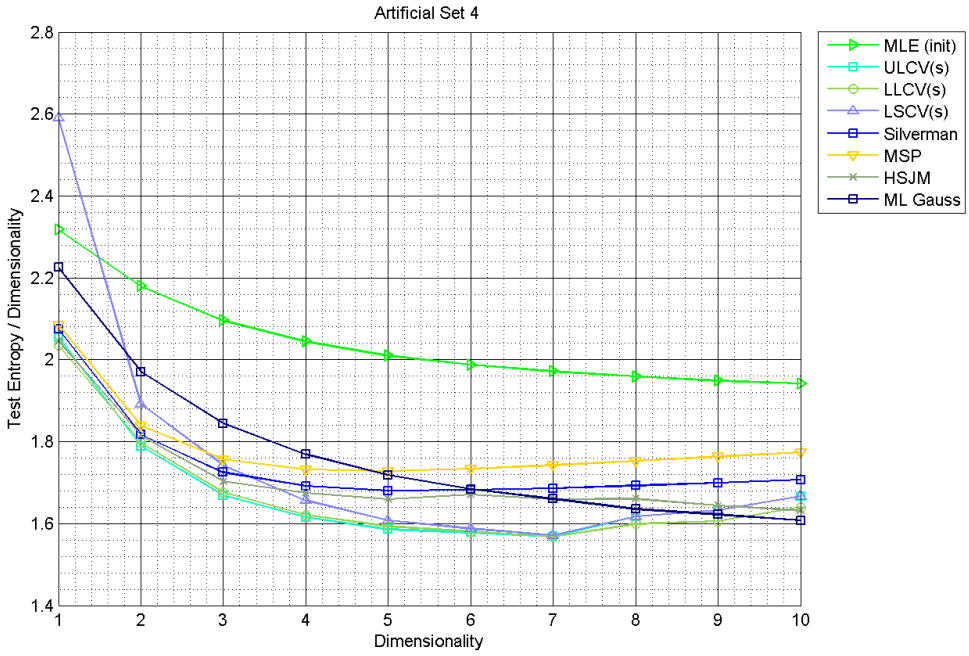

Figure 5.7: *Test entropy results for Artificial Set 4 (D 1-10)*

In *Artificial Set 5*, we introduce intra-feature scale variations to the bimodal data, as explained in the experimental design. Figure 5.9 shows that the LSCV(s) again underperforms compared to the other estimators on one-dimensional data. The MLE(i) estimator consistently performs worse compared to the other estimators, as in the case of Artificial Set 4. This is again to be expected in view of the bimodality of the datasets in Artificial Set 5 that leads to over-smoothing of bandwidths, which is more detrimental in lower dimensions. The HSJM, LSCV(s) and LLCV(s) seem to have the best performance. This is not surprising for the LLCV(s) estimator, since the bandwidths are adapted based on the nearest neighbour distances, which thus accounts for scale variations within features. It is, however, surprising that the HSJM and LSCV(s) perform so well under intra-feature scale variations, since a single bandwidth is estimated per feature across the entire dataset.

Figure 5.10 shows that, similar to our previous observations, the performances of the LSCV and LLCV estimators degrade significantly as dimensionality increases. The ML Gauss estimator performs consistently optimally for dimensionalities of 20 and higher and the Silverman estimator consistently outperforms the MSP estimator. The relative

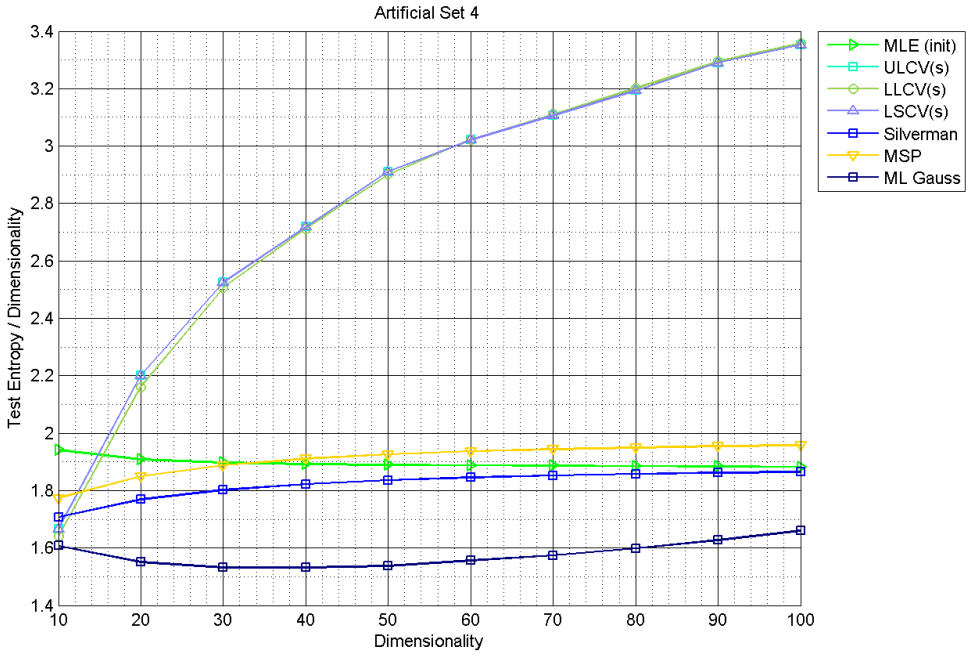

Figure 5.8: *Test entropy results for Artificial Set 4 (D 10-10)*

performance of the MLE(i) estimator, however, seems to improve as dimensionality increases.

In *Artificial Set 6* we introduce correlation between the features of the bimodal data with intra-feature scale variations. Figure 5.11 shows that the LSCV(s) estimator again underperforms for one-dimensional data, and that the MLE(i) estimator consistently underperforms for these low dimensions. The results are very similar to those in Artificial Set 5, and the performance of the ML Gauss estimator improves relative to the other estimators as dimensionality increases. It should be noted that the two modes from which data were sampled have identical covariance matrices, which makes this form of correlation more acceptable for the ML Gauss estimator.

Figure 5.12 shows that the LSCV and LLCV experience severe performance degradation on the high-dimensional data. The ML Gauss estimator consistently performs optimally, the Silverman estimator consistently outperforms the MSP estimator, and the relative performance of the MLE(i) estimator improves as dimensionality increases. It is surprising that the effect of anisotropy on the performance of the ML Gauss estimator does not hold for high dimensions. (This unexpected trend clearly results from the way we have chosen to parameterise our random scale factors and offsets, since very poor ML Gauss estimation can be ensured by choosing widely separated means for the two modes.)

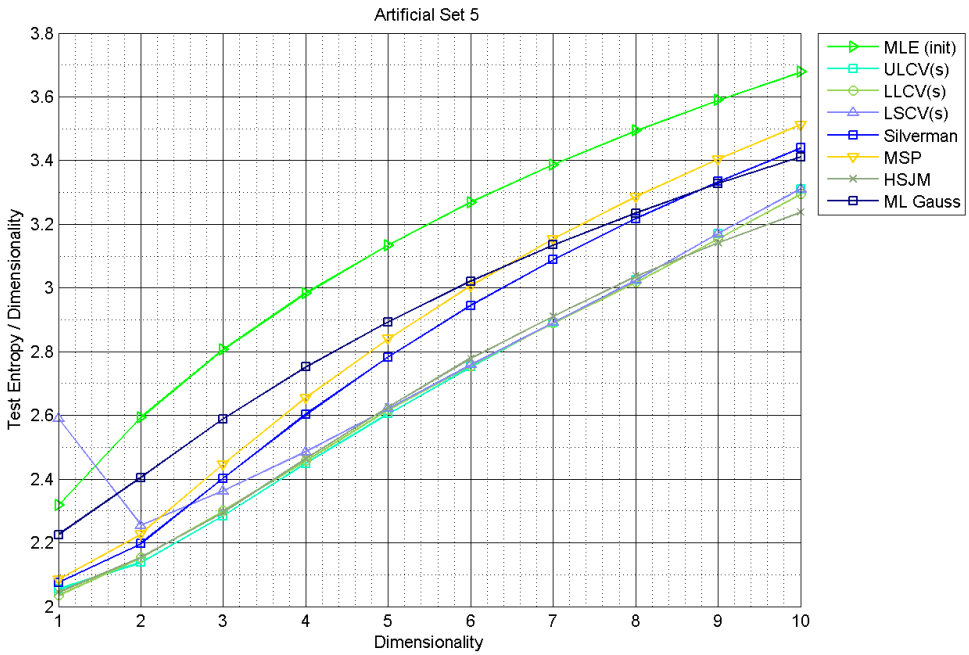

Figure 5.9: *Test entropy results for Artificial Set 5 (D 1-10)*

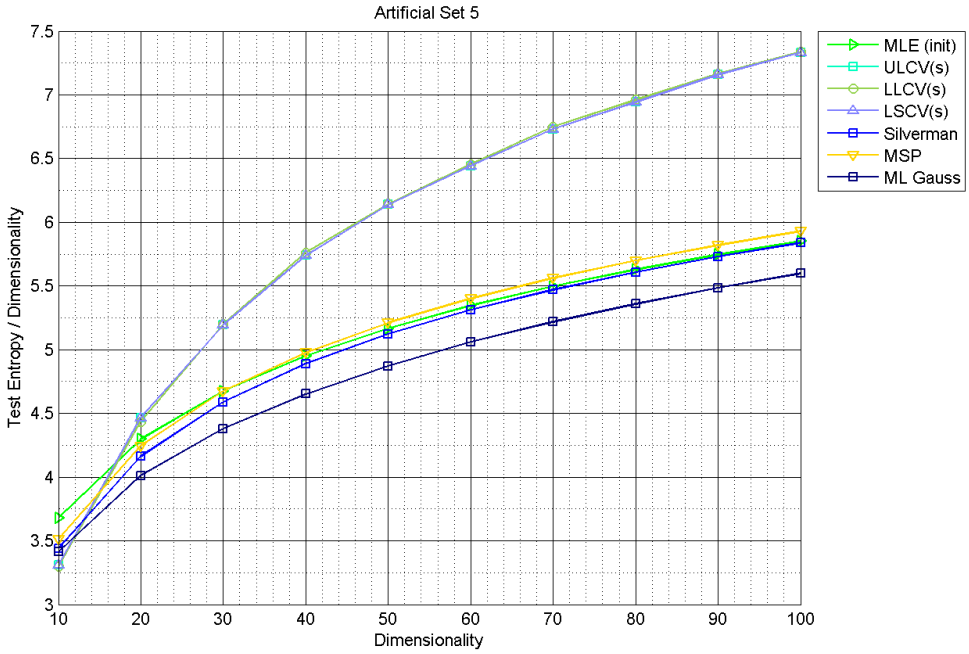

Figure 5.10: *Test entropy results for Artificial Set 5 (D 1-10)*

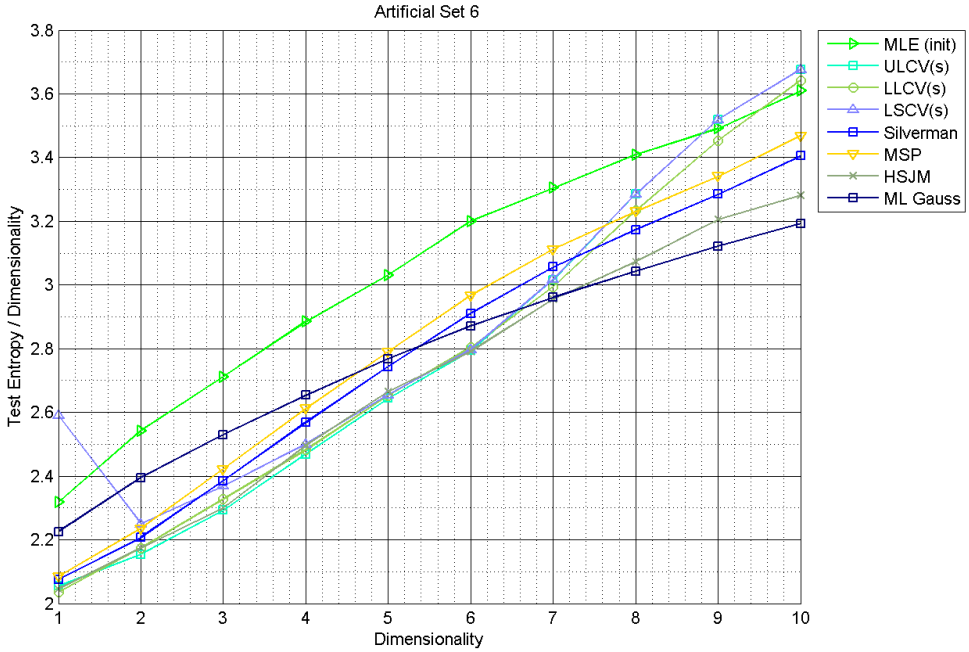

Figure 5.11: *Test entropy results for Artificial Set 6 (D 1-10)*

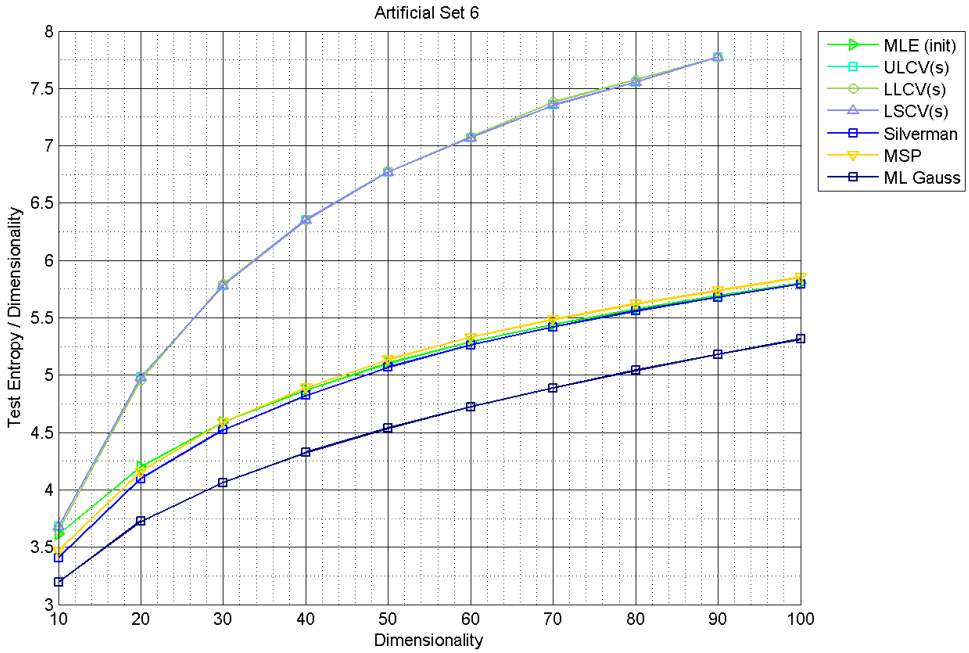

Figure 5.12: *Test entropy results for Artificial Set 6 (D 10-100)*

In *Artificial Set 7* the covariance matrices of the two MVN Gaussian modes are not identical, since the two modes point in opposite directions in the first dimension. Figure 5.13 shows that the relative performances of the estimators are very similar to the results in Artificial Set 6, with the exception of the ML Gauss estimator. The ML Gauss does not outperform the other estimators in dimensions 8-10 as in Artificial Set 6. This is to be expected, since the covariance matrices of the two modes are no longer identical, which makes the unimodal assumption of the ML Gauss estimator less suitable.

Figure 5.14 shows, similar to all our observations for high-dimensional data, that the performances of the LSCV and LLCV estimators degrade significantly as dimensionality increases. The ML Gauss estimator again consistently performs optimally, the Silverman estimator consistently outperforms the MSP estimator, and the relative performance of the MLE(i) estimator improves as dimensionality increases. Again, the effect of anisotropy on the performance of the ML Gauss estimator does not hold for high dimensions, as a consequence of our chosen parameterisation.

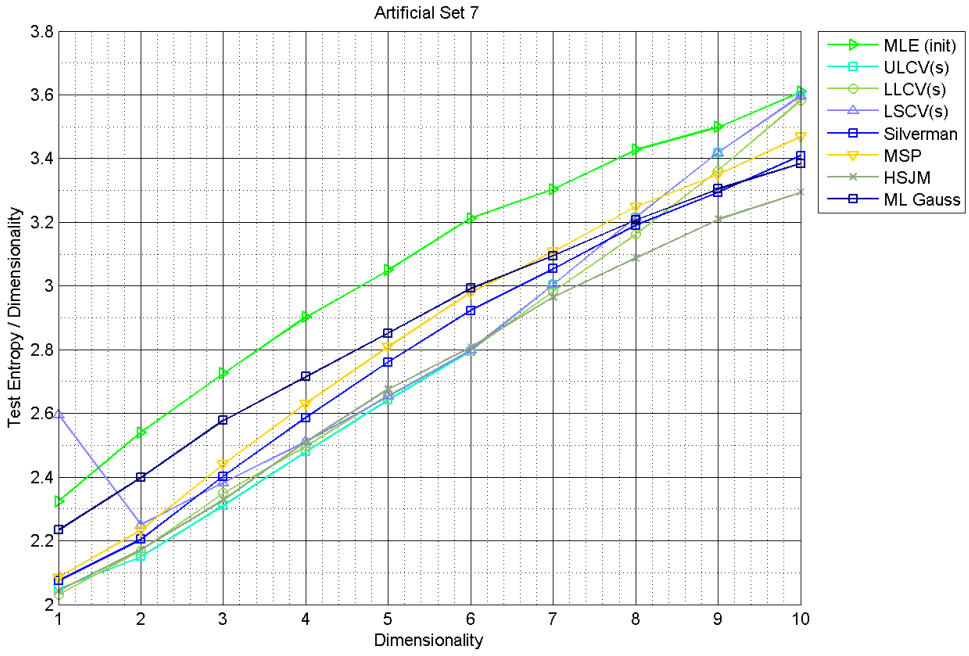

Figure 5.13: *Test entropy results for Artificial Set 7 (D 1-10)*

5.4.2 EXPERIMENT 2

Tables 5.4 and 5.5 show the comparative CV entropy results of the conventional estimators on the CV datasets for class-specific PCA5 and PCA1 pre-processing respectively.

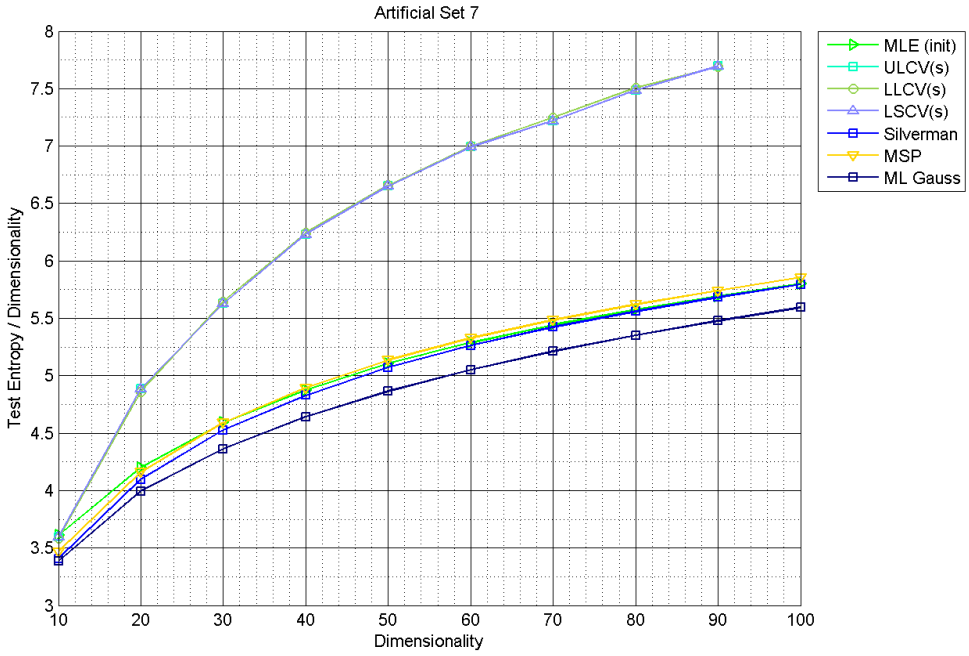

Figure 5.14: *Test entropy results for Artificial Set 7 (D 10-100)*

Note that where the dimensionality of a dataset is less than five, PCA5 will only decorrelate the data and the dimensionality thus remains the same. The dataset name is indicated with "DS", the class number with "C" and the dimensionality of the class after pre-processing with"K". We also abbreviate the dataset names with two letter acronyms, the MLE(i) estimator as "MLE(i)" and the ML Gauss estimator as "Gauss". We make use of colour to visually represent the relative performance of the estimators, where green indicates the lowest entropy for a specific class and red the highest entropy for a specific class.

The *Old Faithful* dataset results in in Table 5.4 show that the ULCV, LLCV and HSJM estimators perform competitively on the two-dimensional dataset, and that the MLE(i) and ML Gauss estimators under-perform.

The *Balance Scale* dataset results in Table 5.4 show that the Silverman, MSP and HSJM estimators perform competitively on all classes of the four-dimensional dataset. Interestingly, the excellent performance of these estimators on the PCA5 dataset for class 2 does not hold for the PCA1 results in Table 5.5, although they still perform competitively on the PCA data. We observe that for the PCA1 dataset, class 2 only has three dimensions. This implies that the fourth dimension (that has been dropped in PCA1), might have contained useful information for the purpose of density estimation. We also note that for the class 2 PCA5 data, the ULCV and LSCV estimators

Table 5.4: CV entropy results (class-specific PCA5)

DS	C	K	MLE(i)	ULCV(s)	LLCV(s)	LSCV(s)	Silverman	MSP	HSJM	Gauss
OF	1	2	2.1619	1.5114	1.5190	1.6933	1.6418	1.7041	1.5120	2.0179
BS	1	4	5.7768	6.0706	6.4392	6.0706	5.4861	5.4534	5.4718	5.5393
	2	4	3.8108	10.9974	10.9706	10.9974	-28.7634	-28.8503	-27.7191	4.2203
	3	4	5.7672	6.0643	6.4294	6.0643	5.4947	5.4553	5.4409	5.5254
IR	1	4	5.5010	5.7345	6.7486	5.7581	5.6587	5.5549	7.1666	5.6510
	2	4	4.8085	4.9665	4.9535	4.9731	4.8840	4.7888	5.3478	4.6654
	3	4	5.0323	4.9976	5.2579	5.1485	5.0947	5.0032	5.5725	4.9727
DB	1	5	7.6345	7.2352	8.3182	7.2195	7.2378	7.2376	8.7124	7.6905
	2	5	7.7473	7.4508	7.6156	7.4624	7.4759	7.4759	8.3736	7.6592
HS	1	5	8.3855	8.1854	8.3171	8.1928	8.1676	8.1849	8.9343	8.2867
	2	5	8.3709	8.2130	8.4184	8.2571	8.3159	8.2435	9.4678	8.2640
VC	1	5	9.0967	8.5931	8.4109	8.5931	8.4350	8.5600	8.5136	8.9478
	2	5	8.9493	8.7049	8.5827	8.7313	8.3850	8.4799	8.5244	8.7638
	3	5	8.7628	7.7522	11.6123	7.8590	7.6507	7.6942	8.7665	9.5198
	4	5	9.1234	8.2437	8.3692	8.2256	8.1073	8.2510	8.2330	9.4099
WF	1	5	8.5599	8.2734	8.2803	8.4415	8.2850	8.2736	8.8006	8.2073
	2	5	8.4912	8.2186	8.2510	8.3847	8.2345	8.2189	8.6721	8.1328
	3	5	8.4642	8.1953	8.2139	8.3495	8.2147	8.1956	8.7014	8.1192
IS	1	5	9.7102	9.7017	9.8104	20.9404	10.1500	9.8599	15.9452	9.8734
	2	5	10.3325	8.9376	9.5393	9.1564	9.0250	9.1386	10.5080	10.7312

have identical results. Since both estimators are initialised with identical bandwidths, the optimal entropy results of these estimators are identical if both estimators have higher entropy values when the bandwidths are updated iteratively – for such cases, the estimators thus do not improve on the initial entropy score. The optimal entropy score is thus the entropy score of the Silverman bandwidth estimate.

The *Iris* dataset results in Tables 5.4 and 5.5 show that the PCA5 and PCA1 transformations reduce the dataset to the same intrinsic dimensionality; the results are thus identical. We observe in these tables that the ML Gauss, MLE(i), Silverman and MSP estimators perform competitively on the four-dimensional Iris dataset. The HSJM estimator consistently underperforms on all three classes.

The *Diabetes* dataset results in Table 5.4 show that the ULCV, LSCV, Silverman and MSP estimators perform competitively on the five–dimensional dataset. The results in Table 5.2 show that the performance of the Silverman and MSP estimators remains competitive on the eight-dimensional dataset and that the ULCV and LSCV experience degradation in performance (relative to the other datasets). The LLCV, on the other hand, experiences a relative increase in performance as the dimensionality increases from 5 to 8. The HSJM, MLE(i) and ML Gauss estimators underperform

Table 5.5: CV entropy results (class-specific PCA1)

DS	C	K	MLE(i)	ULCV(s)	LLCV(s)	LSCV(s)	Silverman	MSP	HSJM	Gauss
OF	1	2	2.1619	1.5114	1.5190	1.6933	1.6418	1.7041	1.5120	2.0179
BS	1	4	5.7768	6.0706	6.4392	6.0706	5.4861	5.4534	5.4718	5.5393
	2	3	5.0003	4.9890	5.1079	5.1893	4.8653	4.8488	4.8127	4.8764
	3	4	5.7672	6.0643	6.4294	6.0643	5.4947	5.4553	5.4409	5.5254
IR	1	4	5.5010	5.6716	6.7132	5.7023	5.6587	5.5549	7.1666	5.6510
	2	4	4.8085	4.9665	4.9535	4.9731	4.8840	4.7888	5.3478	4.6654
	3	4	5.0323	4.9976	5.2579	5.1485	5.0947	5.0032	5.5725	4.9727
DB	1	8	10.4242	10.4729	9.9768	10.4729	10.0000	9.9474	11.3712	10.7987
	2	8	10.8681	10.6695	10.5866	10.6695	10.5426	10.5469	11.1479	11.0914
HS	1	13	18.3467	23.7675	24.1119	23.7675	18.6033	18.4294	18.3579	18.4897
	2	13	17.8388	18.8418	21.4540	20.4495	17.7807	17.7338	17.9350	18.3348
VC	1	9	11.5707	16.1516	15.4709	16.1516	10.9220	11.0842	10.9019	11.5029
	2	8	10.8556	15.1835	14.5981	15.1835	10.2151	10.3520	10.2152	10.7001
	3	8	10.8859	12.6065	20.9062	12.6065	9.3708	9.5106	10.1298	11.8372
	4	11	12.7546	14.7682	14.8125	14.8003	12.1415	12.2842	12.2024	14.1832
WF	1	21	26.5689	47.9183	46.4132	47.9183	27.1867	26.7686	38.3311	24.9595
	2	21	29.0202	44.9225	46.4381	44.9225	29.6012	29.2137	38.9844	27.4697
	3	21	28.9282	46.9280	49.5978	46.9280	29.5804	29.1566	40.6167	27.3635
IS	1	33	42.2876	49.9981	50.0368	58.4048	46.9883	44.0018	60.6990	52.1698
	2	12	14.9079	17.6331	18.4547	17.6331	18.0641	16.6253	25.3506	17.1796

consistently on both the five-and eight-dimensional datasets. We also observe an increase in entropy between the results for PCA5 and PCA1, which is due to the higher dimensionality of the PCA1 dataset.

The *Heart(Statlog)* dataset results in Table 5.4 show that the ULCV, LSCV, Silverman and MSP estimators, again, perform competitively on the five-dimensional dataset. The results in Table 5.5 show that the performance of the Silverman and MSP estimators remains competitive on the 13-dimensional dataset and that the ULCV and LSCV again experience a relative degradation in performance. The LLCV estimator underperforms on both datasets, and also experiences degradation in relative performance as dimensionality increases. The MLE(i)and HSJM estimators, on the other hand, experience a relative improvement in performance as dimensionality increases, and performs competitively on the 13-dimensional dataset. The relative performance of the ML Gauss estimator remains similar, and this estimator does not perform competitively on both the PCA5 and PCA1 datasets.

The *Vehicle* dataset results in Tables 5.4 and 5.5 show that the Silverman and MSP estimators perform competitively on the five-dimensional PCA5 dataset and on the higher-dimensional PCA1 dataset. The HSJM estimator experiences a relative

increase in performance as dimensionality increases, and performs competitively on the PCA1 dataset. The MLE(i), ULCV, LSCV and ML Gauss estimators underperform on most classes of the PCA5 and PCA1 datasets.

The *Waveform* dataset results in Tables 5.4 and 5.5 show that the ML Gauss estimator performs optimally across all classes on both the five-dimensional PCA5 and 21-dimensional PCA1 datasets. The Silverman and MSP estimators perform competitively across all classes, and the HSJM estimator consistently underperforms on the PCA5 and PCA1 datasets. The ULCV, LLCV and LSCV estimators consistently underperform across all classes, and experience a drastic degradation in relative performance as the dimensionality increases from 5 to 21. Conversely, the relative performance of the ML Gauss and MLE(i) estimators increase drastically as dimensionality increases.

The *Ionosphere* dataset results in Table 5.4 show that the ULCV, Silverman and MSP estimators perform competitively across all classes on the five-dimensional PCA5 dataset. We see that the LSCV underperforms severely on class 1 of the PCA5 dataset, and that a valid bandwidth estimate was not obtained for the 33-dimensional class 1 of the PCA1 dataset in Table 5.2. The MLE(i) estimator experiences a drastic improvement in relative performance across both classes as dimensionality increases. We also note that the entropy scores of class 1 on the PCA1 dataset are much higher than the entropy scores of class 2 on the PCA dataset because of the large difference in dimensionality (33 as opposed to 12).

In summary, we have observed in this experiment that the Silverman and MSP estimators perform consistently well across all dimensions, and that the relative performances of ML-Gauss and MLE(i) estimators increase with dimensionality. We also observed that the ULCV, LLCV and LSCV estimators generally did not perform well on dimensionalities of 9 and higher, and that the HSJM estimator significantly underperformed on the high-dimensional PCA1 Waveform and Ionosphere datasets.

We report on the results of the global PCA5 and PCA1 datasets in Appendix A, where we make general comments regarding the differences between the class-specific and global PCA results.

5.4.3 EXPERIMENT 3

Table 5.6: Letter test entropy results (class-specific PCA5)

C	K	NC	MLE(i)	ULCV(s)	LLCV(s)	LSCV(s)	Silverman	MSP	HSJM	Gauss
1	5	789	8.5010	6.1808	6.2032	6.1596	6.8899	7.1586	6.4662	8.8204
2	5	766	9.2746	7.5052	7.8272	7.9781	7.9753	8.1984	7.9463	9.4780
3	5	736	9.0447	7.1479	7.1755	7.2868	7.7186	7.9532	7.3644	9.1031
4	5	805	8.6534	6.7295	6.7326	7.2920	7.2326	7.4644	7.6237	8.9201
5	5	768	8.8843	6.5675	6.6839	7.2536	7.3937	7.6787	6.7275	8.9505
6	5	775	8.9909	6.9747	7.6123	7.0571	7.5746	7.8235	7.2301	8.9826
7	5	773	8.9098	7.0713	7.1257	7.0716	7.6217	7.8498	7.4057	8.8665
8	5	734	8.7473	6.6398	6.7005	6.8395	7.1379	7.3701	7.5861	8.9307
9	5	755	8.7662	5.8551	5.9796	6.7905	7.0534	7.3922	5.9927	8.9958
10	5	747	9.1996	6.6060	6.8621	6.6055	7.6609	7.9615	6.6198	9.2554
11	5	739	8.9522	6.8819	6.8975	6.9326	7.5672	7.8222	7.2443	9.0397
12	5	761	8.8066	5.9326	6.7099	5.9494	7.2057	7.5336	5.9771	8.9155
13	5	792	8.6851	6.0365	6.1745	6.0921	7.0623	7.3464	6.0944	8.8657
14	5	783	8.9441	6.3179	6.7553	6.8151	7.2403	7.5297	6.4774	9.3321
15	5	753	8.8595	7.6466	7.6545	8.0578	7.8057	7.9529	8.5382	9.0477
16	5	803	9.1070	7.1168	7.4353	7.2499	7.7852	8.0293	7.3492	9.0725
17	5	783	8.7441	6.8391	6.9470	6.8556	7.2906	7.5207	7.4603	9.0645
18	5	758	9.0200	7.1711	7.1989	7.2538	7.7392	7.9863	7.5312	9.0243
19	5	748	8.9538	6.5028	6.4594	6.4896	7.6016	7.8947	6.3888	8.8047
20	5	796	9.1107	6.6207	7.0889	6.6262	7.5966	7.8883	6.6370	9.0690
21	5	813	8.7536	6.4467	6.5019	7.1448	7.1924	7.4615	7.1288	8.8848
22	5	764	8.9768	6.7560	6.9498	6.7704	7.5578	7.8194	6.8375	9.1319
23	5	752	9.2657	7.1362	7.5099	7.4382	7.7438	8.0064	7.6131	9.5949
24	5	787	8.8773	6.6471	7.1030	7.0633	7.2540	7.5080	7.0525	9.1758
25	5	786	8.8556	6.2757	6.4317	6.2740	7.3537	7.6512	6.3087	8.7377
26	5	734	9.0164	6.9555	6.9810	8.9270	7.5629	7.8185	7.5565	8.9785

Table 5.7: Letter test entropy results (class-specific PCA1)

C	K	NC	MLE(i)	ULCV(s)	LLCV(s)	LSCV(s)	Silverman	MSP	HSJM	Gauss
1	15	789	13.2926	19.9349	18.0448	19.9349	11.2248	11.9195	10.6207	15.1444
2	15	766	14.7607	17.6263	18.2540	17.6263	13.2300	13.7460	12.8959	16.0683
3	15	736	14.1119	16.4159	15.7291	16.4159	12.5233	13.1484	12.1285	15.1427
4	14	805	14.1077	14.4740	13.2003	14.4740	12.2357	12.8792	11.9151	15.4898
5	15	768	13.1069	16.9629	15.5996	16.9629	11.2543	11.9690	10.5711	14.2677
6	13	775	13.1498	13.1674	13.1961	13.1674	11.1886	11.8443	10.5422	14.2545
7	15	773	14.9531	21.5078	20.7061	21.5078	13.2091	13.8168	12.9103	16.4409
8	14	734	14.6584	15.6469	14.2643	15.6469	12.0327	12.7703	11.5800	16.5885
9	13	755	13.5458	13.7429	14.8962	13.7429	10.9446	11.7310	10.8178	16.1792
10	13	747	12.4767	13.6645	12.4283	13.6645	10.3277	11.0836	9.3387	13.8763
11	15	739	13.6784	17.2278	16.6712	17.2278	11.8128	12.5628	11.2306	15.2347
12	12	761	11.5495	11.9264	12.1383	11.9264	9.6349	10.3763	8.6058	12.8659
13	14	792	13.7735	13.3229	12.8881	13.3229	11.1204	12.0003	9.9798	15.6487
14	14	783	13.7528	16.0171	17.9213	16.0171	11.4153	12.0858	11.0703	16.0152
15	16	753	15.2257	18.7668	18.1951	18.7668	13.9470	14.4486	13.6313	16.8445
16	15	803	13.9270	19.8545	17.6055	19.8545	12.1621	12.8541	11.6101	15.1049
17	15	783	14.1865	20.0228	17.8473	20.0228	12.4594	13.0680	12.0017	15.7706
18	15	758	14.3232	20.1125	19.7045	20.1125	12.6791	13.3217	12.2919	15.3979
19	15	748	13.6627	23.9613	24.7224	23.9613	11.7087	12.6267	10.7761	14.8713
20	13	796	12.0510	15.8027	15.3266	15.8027	10.1052	10.7146	9.9396	13.3641
21	13	813	13.1536	13.9737	13.2000	13.9737	10.8012	11.4191	11.3568	14.6759
22	14	764	13.6603	16.1495	16.4227	16.1495	11.9660	12.6572	11.5475	14.9304
23	15	752	14.5485	16.5111	17.8693	16.5111	12.4537	13.1731	12.0028	16.6306
24	15	787	14.4447	14.7889	14.0680	14.7889	12.5252	13.1195	12.4355	16.2315
25	14	786	13.1325	16.7760	14.5764	16.7760	10.8837	11.6950	10.2508	14.8458
26	15	734	14.0939	17.0518	16.6731	17.0518	12.0371	12.6910	12.6700	16.1451

The *Letter* PCA5 dataset results in Table 5.6 show that the ULCV estimator performs optimally on all classes (with the exception of class 19, where the performance is competitive). The LLCV and LSCV estimators perform competitively on most classes, and the Silverman, MSP and HSJM estimators perform moderately on most classes. The MLE(i) and Gauss estimators, however, underperform on all classes.We observe converse estimator performance behaviour for the ULCV, LLCV and LSCV estimators on the higher-dimensional PCA1 results shown in Table 5.4. These estimators underperform on most classes, while the Silverman and HSJM estimators perform competitively. The relative performance of the ML Gauss estimator increases with the increase in dimensionality, but still remains sub-optimal.

The *Segmentation* dataset results in Tables 5.8 and 5.9 show that the Silverman and MSP estimators perform competitively on both the five-dimensional PCA5 dataset

Table 5.8: Segmentation test entropy results (class-specific PCA5)

C	K	NC	MLE(i)	ULCV(s)	LLCV(s)	LSCV(s)	Silverman	MSP	HSJM	Gauss
1	5	330	8.9567	8.6571	9.4309	8.6400	8.6097	8.5353	10.5575	9.1783
2	5	330	8.6969	7.9876	7.7229	8.8720	7.7100	7.7971	10.6039	9.2186
3	5	330	8.1240	6.6543	8.9703	10.8726	7.0207	7.0412	13.8225	9.0371
4	5	330	8.1394	8.7773	10.0652	14.5373	11.0827	10.0475	26.0267	10.0415
5	5	330	7.8425	7.7285	16.3031	28.9613	8.0964	7.7062	25.3649	8.9870
6	5	330	8.6377	8.1480	8.2896	7.9676	8.0393	8.0272	9.1536	8.6815
7	5	330	7.2834	4.8831	5.0227	4.7681	5.6182	5.9123	4.8821	7.3298

Table 5.9: Segmentation test entropy results (class-specific PCA1)

C	K	NC	MLE(i)	ULCV(s)	LLCV(s)	LSCV(s)	Silverman	MSP	HSJM	Gauss
1	13	330	15.8936	19.7828	19.7140	19.7828	15.2068	15.0066	16.0416	19.4171
2	11	330	14.0655	16.0282	19.8177	16.0282	12.9305	13.0771	15.7156	15.6992
3	12	330	12.4763	10.1055	13.9903	26.2003	10.5969	10.8379	20.6875	15.7573
4	11	330	12.9607	13.8961	14.1103	13.8961	17.5817	15.9097	11.6879	18.0036
5	11	330	12.1976	14.8880	28.4377	16.5385	16.4508	14.8832	56.2277	17.1998
6	11	330	13.3608	18.1200	16.0733	18.1200	11.3063	11.5558	11.2357	14.8150
7	10	330	10.1812	9.8896	11.5070	9.8896	8.5376	8.8146	11.4790	11.1225

and the higher-dimensional PCA1 dataset. Again, the ULCV, LLCV and LSCV estimators experience a relative decrease in performance when the dimensionality of the data increases. The MLE(i) estimator, on the other hand, again experiences an increase in relative performance as dimensionality increases. The ML Gauss estimator underperforms on both the PCA5 and PCA1 datasets.

Table 5.10: Landsat test entropy results (class-specific PCA5)

C	K	NC	MLE(i)	ULCV(s)	LLCV(s)	LSCV(s)	Silverman	MSP	HSJM	Gauss
1	5	1533	9.1219	8.4065	8.5952	8.4809	8.5251	8.5239	10.8315	9.1363
2	5	703	10.0996	8.1881	8.1793	8.2670	8.5457	8.7830	9.8281	10.4275
3	5	1358	9.9283	9.4269	11.3001	9.4937	9.5859	9.4692	12.6314	10.2360
4	5	626	9.5682	8.6614	10.1450	8.7826	8.5052	8.5341	10.7932	9.9060
5	5	707	10.3927	9.5970	9.5845	9.7385	9.6460	9.7217	10.9622	10.5202
6	5	1508	9.6239	9.0323	12.6993	9.2426	9.2208	9.0951	14.0773	9.9167

Table 5.11: Landsat test entropy results (class-specific PCA1)

C	K	NC	MLE(i)	ULCV(s)	LLCV(s)	LSCV(s)	Silverman	MSP	HSJM	Gauss
1	8	1533	11.6149	11.3411	10.9108	11.3411	10.9399	10.8943	13.9897	11.8245
2	10	703	13.8418	13.3700	12.4680	13.3700	12.3249	12.5838	14.8169	14.8997
3	14	1358	17.1282	26.3911	38.3289	26.3911	17.1899	16.7990	16.8388	18.0135
4	13	626	15.9430	21.6951	34.4233	21.6951	14.7206	14.7212	14.8199	16.8540
5	11	707	15.4558	18.1305	20.7268	18.1305	14.7804	14.8053	15.6636	16.2852
6	11	1508	15.0525	16.4696	28.7230	16.4696	15.1227	14.6888	17.6072	16.0888

The *Landsat* dataset results in Tables 5.10 and 5.11 show that the ULCV estimator performs optimally on the PCA5 dataset, and that the relative performance of the estimator decreases for the higher-dimensional PCA1 dataset. The LLCV and LSCV again experience a relative decrease in performance as dimensionality increases, and conversely the MLE(i) experiences a relative increase in performance as dimensionality increases. The Silverman and MSP estimators perform competitively on both the PCA5 and PCA1 datasets, and also experience a relative performance increase with an increase in dimensionality. The HSJM estimator underperforms on all classes on the PCA5 dataset, and experiences a relative increase in performance in some classes for the PCA1 dataset. The ML Gauss estimator underperforms on both the PCA5 and PCA1 datasets.

The *Optdigits* dataset results in Tables 5.12 and 5.13 show that the ULCV and LSCV estimators perform competitively on the PCA5 data for most classes, and underperform for most classes of the higher-dimensional PCA1 dataset. The LLCV estimator performs competitively on some classes on the PCA5 dataset, and underperforms on all classes for the PCA1 dataset. The Silverman and MSP estimators again perform competitively on most classes for both the PCA5 and higher-dimensional PCA1 datasets. The HSJM and ML Gauss estimators experience a relative increase in performance as dimensionality increases. The HSJM estimator underperforms on the PCA5 dataset

Table 5.12: Optdigits test entropy results (class-specific PCA5)

C	K	NC	MLE(i)	ULCV(s)	LLCV(s)	LSCV(s)	Silverman	MSP	HSJM	Gauss
1	5	554	10.8597	10.4546	11.4455	10.4511	10.4373	10.4590	11.4862	10.7261
2	5	571	11.2238	9.7869	11.3589	10.0520	9.9005	10.0060	12.2221	11.9799
3	5	557	11.1693	10.6538	10.8321	10.6852	10.7486	10.7175	12.6983	11.3608
4	5	572	10.8052	10.7150	14.7925	10.6734	10.7458	10.6480	12.5776	10.9551
5	5	568	10.3482	9.4964	8.8947	8.7507	9.0864	9.2828	9.2950	10.5059
6	5	558	10.9935	10.3020	10.4525	10.3047	10.3475	10.3837	11.4935	10.9721
7	5	558	10.8467	10.8575	12.3872	11.1234	11.6699	11.3303	16.1433	11.0407
8	5	566	11.0645	10.8667	11.0906	10.8023	10.7893	10.7510	13.4742	10.9401
9	5	554	11.1262	10.8674	10.8646	10.9231	10.8886	10.8482	11.5301	10.8957
10	5	562	10.9813	10.3198	10.3184	10.3495	10.3603	10.4193	11.0516	11.0665

and performs moderately on the PCA1 dataset. The ML Gauss estimator performs moderately on most classes of the PCA5 dataset and performs competitively on the PCA1 dataset.

Table 5.13: Optdigitstest entropy results (class-specific PCA1)

C	K	NC	MLE(i)	ULCV(s)	LLCV(s)	LSCV(s)	Silverman	MSP	HSJM	Gauss
1	46	554	53.8426	140.2504	162.1402	140.2504	55.0121	53.3564	121.8120	52.9456
2	36	571	40.1827	75.0960	102.7028	75.0960	40.0183	39.9282	46.1139	45.7599
3	41	557	55.7362	117.7803	140.3123	117.7803	61.1132	57.3356	90.9314	60.4556
4	50	572	63.4517	142.2787	183.0795	142.2787	69.7068	65.8302	153.2374	78.1272
5	39	568	45.2993	95.3545	92.8862	95.3545	45.1332	45.2913	60.2590	48.6055
6	46	558	57.0661	130.0414	136.8794	130.0414	58.9782	57.0790	112.0333	58.9781
7	37	558	46.9708	97.5448	124.2272	97.5448	46.9650	47.9311	78.2761	50.1364
8	41	566	53.4834	105.5003	121.7572	105.5003	55.9758	53.7137	77.1611	57.4866
9	50	554	62.8508	123.9320	124.8377	123.9320	64.2079	62.0098	96.3983	62.6491
10	43	562	57.4619	113.7416	124.5037	113.7416	56.5346	59.7310	99.0382	60.1219

The *Isolet* PCA5 dataset results in Table 5.14 show that the ULCV, LSCV, Silverman and MSP estimators perform competitively on most classes of the five-dimensional dataset, and that the HSJM estimator generally underperforms. The PCA1 results in Table 5.15 show that the ULCV, LLCV and LSCV estimators consistently underperform on the high-dimensional data and that the MLE(i), Silverman and MSP estimators perform competitively on most classes, compared to the ML Gauss estimator that consistently performs optimally on all classes. We also observe that the HSJM estimator does not give meaningful results for 13 of the 26 classes. The bandwidths of these classes were investigated, and we found that all bandwidths for these classes were very large (in the order of 10^4), thus the likelihood scores of all test samples are 0 owing

Table 5.14: Isolet test entropy results (class-specific PCA5)

C	K	NC	MLE(i)	ULCV(s)	LLCV(s)	LSCV(s)	Silverman	MSP	HSJM	Gauss
1	5	300	16.3464	15.9057	15.9305	16.0074	15.9339	15.9987	16.2760	15.9659
2	5	300	15.8114	15.2251	18.3920	15.2241	15.2733	15.2433	16.3991	15.6929
3	5	300	16.5776	16.3727	16.8537	16.2808	16.2951	16.3109	17.0933	16.1481
4	5	300	16.2090	15.8296	17.3434	15.8353	15.8267	15.8636	16.5559	15.8717
5	5	300	16.3759	16.0439	16.0694	16.1243	16.0378	16.0765	16.4481	16.0460
6	5	300	16.1510	15.4689	15.6772	15.4689	15.3722	15.4680	16.0273	16.0258
7	5	300	16.1280	15.6802	15.7055	15.5995	15.5534	15.6669	15.7452	15.7445
8	5	300	16.3568	15.8340	16.5158	15.9362	15.8135	15.9158	15.9893	16.0152
9	5	300	16.4492	17.2587	17.6673	16.7972	17.3416	17.0154	20.1168	16.6718
10	5	300	16.0164	15.2372	15.3291	15.2178	15.2038	15.3194	15.4599	15.7890
11	5	300	16.2246	15.9499	15.9935	15.9499	15.9712	15.9801	17.2820	16.3152
12	5	300	16.4872	17.4557	16.8878	17.3693	17.7369	17.3147	21.1982	17.0577
13	5	300	15.7348	14.9685	14.9230	15.0245	15.0318	15.1681	15.0796	15.5537
14	5	300	15.7358	15.4938	15.3305	15.2028	15.0641	15.0627	16.1588	15.6874
15	5	300	15.7739	15.4970	15.4083	15.4196	15.6394	15.5821	17.5251	17.5199
16	5	300	15.8505	15.0747	16.1217	15.0820	15.1094	15.1591	15.6495	15.7309
17	5	300	16.1152	15.8602	15.8457	15.8594	15.8228	15.8669	16.3323	15.5785
18	5	300	16.6431	16.4367	16.3174	16.3223	16.3284	16.3198	17.0791	16.2938
19	5	300	16.2124	16.2619	16.7708	16.4205	16.9965	16.6701	19.2783	16.0424
20	5	300	16.0753	15.4843	15.5004	15.5234	15.5288	15.6273	15.8112	15.6931
21	5	300	15.7672	14.9134	14.9830	15.1015	15.0450	15.1949	14.9224	15.4109
22	5	300	16.4810	16.3396	16.2952	16.2391	16.2573	16.2285	17.4357	16.0978
23	5	300	16.5390	15.1784	15.5481	15.1917	15.3991	15.5940	15.4764	16.5725
24	5	300	16.0764	15.7645	15.7892	15.7645	15.7648	15.7637	16.4547	15.7773
25	5	300	16.2465	15.8996	15.9449	15.9193	15.8708	15.8948	16.6089	15.8751
26	5	300	16.5589	16.5905	17.4176	16.4409	16.4852	16.4229	17.6562	16.1449

to the large bandwidths, and consequently valid entropy scores cannot be calculated for these classes. Note that the intrinsic dimensionalities of the classes in the PCA1 dataset are very high; the average dimensionality is approximately 103, which explains why the HSJM estimator does not find a valid bandwidth on 13 of the 26 classes, and why most estimators struggle to find suitable bandwidths for classes 9, 12 and 17.

Table 5.15: Isolet test entropy results (class-specific PCA1)

C	K	NC	MLE(i)	ULCV(s)	LLCV(s)	LSCV(s)	Silverman	MSP	HSJM	Gauss
1	92	300	187.3541	375.7138	387.2360	375.7138	190.8883	188.9199	643.0318	176.0779
2	107	300	206.4788	454.5484	484.0007	454.5484	212.5131	210.5833	NaN	188.5294
3	98	300	197.1934	391.7476	411.6608	391.7476	202.9675	200.5494	633.2168	181.4612
4	101	300	200.7884	390.2165	417.8238	390.2165	207.8165	205.1302	636.6129	189.0299
5	103	300	214.7763	410.1299	422.4844	410.1299	218.9028	214.2833	649.6709	201.1150
6	100	300	197.4903	397.4712	417.3551	397.4712	200.6515	198.9031	663.2106	182.3910
7	90	300	183.4525	377.4078	378.5666	377.4078	188.2732	186.6014	683.4447	171.1568
8	88	300	177.7234	361.6492	379.6527	361.6492	182.0967	180.7303	517.8069	165.9693
9	108	300	232.5091	463.5239	471.7874	463.5239	226.6732	232.2718	NaN	207.4642
10	85	300	180.4786	336.7212	339.9663	336.7212	184.6028	181.7460	334.2881	172.9212
11	89	300	185.8624	357.5171	351.0308	357.5171	190.9604	188.3001	593.8495	177.9836
12	118	300	247.0241	496.0116	502.4767	496.0116	250.0967	243.5696	NaN	239.5606
13	123	300	244.0540	518.7484	512.9274	518.7484	249.8319	245.1665	NaN	225.1900
14	119	300	235.7570	491.6797	487.4691	491.6797	241.8353	237.7011	NaN	214.9670
15	103	300	209.4080	415.0969	425.4027	415.0969	213.1697	210.0544	NaN	199.9210
16	104	300	200.9151	399.1593	425.1918	399.1593	207.3090	205.6443	626.9011	184.8603
17	108	300	226.0176	440.4785	444.7663	440.4785	227.7496	223.3443	NaN	209.0767
18	106	300	218.1320	436.1860	434.6972	436.1860	221.2660	217.4032	NaN	203.9322
19	96	300	199.3624	397.9212	404.6384	397.9212	203.7217	200.2767	677.9093	184.4343
20	100	300	196.5860	403.8593	400.5862	403.8593	202.6541	201.3852	NaN	180.6759
21	121	300	235.8053	530.3606	523.0984	530.3606	244.1695	239.9305	NaN	214.3276
22	113	300	223.8469	470.3953	479.0622	470.3953	230.1947	226.4570	NaN	203.2678
23	104	300	203.8252	421.6485	423.3247	421.6485	209.0204	207.2332	NaN	187.6081
24	92	300	196.2914	365.6939	386.3800	365.6939	197.9059	194.6568	523.0979	185.3459
25	109	300	213.5672	448.5116	452.7414	448.5116	221.0781	218.0630	NaN	196.0630
26	100	300	201.6605	399.3444	399.1497	399.3444	206.7172	204.5203	460.3485	185.5410

5.5 CONCLUSION

In this chapter we compared the performance of conventional density estimators and the MLE(i) estimator on a representative set of artificial, CV and RW datasets. The artificial data experiments were very informative, since the properties of the artificial data were known and allowed us to investigate the behaviour of the estimators under different conditions.

On the unimodal artificial data we observed that the ML Gauss estimator generally performed optimally and that the introduction of correlation and scale variations did not adversely affect the performance of the ML Gauss estimator. This was expected, since a full covariance matrix is capable of modelling correlation and scale for unimodal data. The ULCV, Silverman and MSP and MLE(i) estimators performed well on all dimensions, and the LLCV and LSCV estimators generally performed well on lower dimensions (dimensions of 1-10). The LLCV and LSCV estimators experienced severe performance degradation as dimensionalities increased beyond 10.

On the bimodal artificial data the MLE(i) estimator suffered severe performance degradation in lower dimensions because of the bimodality, but performed well in higher dimensions with improved relative performance as dimensionality increased. The remaining estimators were all competitive in lower dimensions. In higher dimensions (dimensions larger than 10), the ML Gauss estimator generally performed optimally, even though the ML Gauss assumed a unimodal distribution.We know that this superior performance of the ML Gauss estimator might change if the distance between the means of the two modes is increased. The Silverman and MSP estimators performed well on the high-dimensional data, and the Silverman estimator generally outperformed the MSP estimator on bimodal data.

On CV and RW data we observed that the ULCV estimator performed very competitively on the low-dimensional PCA5 datasets, but generally suffered severe performance degradation on the higher-dimensional PCA1 datasets. The ULCV, LLCV and LSCV also experienced severe performance degradation on high-dimensional PCA1 data. The performances of the Silverman and MSP estimators were quite consistent across all dimensionalities and tasks, and were generally very competitive. The MLE(i) and ML Gauss estimators experience relative performance increases with increased dimensionality, and might be favoured in very high-dimensional settings. The performance of the HSJM estimator is more difficult to predict, since it performs competitively on some PCA5 and PCA1 tasks, and on other tasks it underperforms. We note, however, that this estimator did not give meaningful results on the Isolet PCA1 dataset, which had an average intrinsic dimensionality of 103.

In summary, we found that the conventional ULCV, LLCV and LSCV estimators are not suitable for dimensionalities larger than 10, and that the Silverman and MSP estimators consistently performed well across all dimensions for the artificial, CV and RW datasets investigated. We also noted that the HSJM estimator performance was more difficult to predict.

In the next chapter we will compare ML bandwidth estimators that require band-

width initialisation to initialise the bandwidth optimisation procedure. Given the consistent performance of the Silverman rule-of-thumb estimator, and the intuitive theoretical motivation that the Silverman estimator optimises the AMISE (assuming a Gaussian reference distribution), we will use the Silverman estimator as the bandwidth initialisation procedure for our ML estimators. Since the relative performance of the MLE(i) and ML Gauss estimators increased as dimensionality increased, these estimators might be suitable bandwidth initialisation candidates for datasets with dimensionalities higher than those we considered in our experiments.

Chapter Six

Comparison of Maximum-likelihood Kernel Bandwidth Estimators

Contents

6.1 Introduction 96

6.2 Methods And Data 97

6.2.1 Data Sampling Procedure 97

6.2.2 Cross-validation data 97

6.2.3 Real-world data 97

6.2.4 Data Pre-processing 97

6.2.5 Methods 97

6.2.6 Kernel bandwidth initialisation 97

6.2.7 Kernel bandwidth regularisation 99

6.2.8 Selecting the optimal number of training iterations 99

6.2.9 Performance evaluation 100

6.2.10 Statistical difference testing 100

6.3 Experimental Design 102

6.3.1 Experiment 1 . . . 102
6.3.2 Experiment 2 . . . 102
6.3.3 Experiment 3 . . . 103
6.4 Results . . . 103
6.4.1 Experiment 1 . . . 103
6.4.2 Experiment 2 . . . 114
6.4.3 Experiment 3 . . . 118
6.5 Analysis of MLE AND MLE(g) Bandwidths . . . 143
6.6 Conclusion . . . 155

6.1 INTRODUCTION

In Chapter 4, we derived the MLE and global MLE kernel bandwidth estimators from the ML LOUT framework. This approach was motivated by the MLL and ML-LOO ML estimators, since these estimators have shown much promise for nonparametric density estimation of high-dimensional pattern recognition tasks [17, 18]. We provided the proofs of these algorithms in the context of the same framework as the MLE algorithm, and pointed out the simplifying assumptions that were made in the derivations of these estimators.

In this chapter we compare the performance of the MLE, global MLE, MLL, spherical covariance ML-LOO and full covariance ML-LOO estimators, to gain a better understanding of whether the more general derivation of the MLE algorithm will give an improvement in practice.

As in Chapter 5, we investigate the performance of the bandwidth estimators on artificial data with data properties that are varied in a controlled fashion. We also compare the performance of the bandwidth estimators on real-world datasets, and make the same distinction between CV datasets and RW datasets. We do, however, consider two additional factors in the experimental design, namely (1) the selection of initial bandwidths to initialise the bandwidth optimisation procedures of the ML estimators, and (2) the selection of the optimal number of training iterations for the bandwidth optimisation procedures.

We describe the methodology for artificial data generation, the methods that will be compared, data pre-processing, bandwidth initialisation, bandwidth regularisation, selecting the optimal number of training iterations, performance evaluation and statistical difference testing in Section 6.2, and design our experiments for artificial, CV and

RW data simulation studies in Section 6.3. We present the results of the simulation studies in Section 6.4 and conclude on our findings in Section 6.5.

6.2 METHODS AND DATA

6.2.1 DATA SAMPLING PROCEDURE

We make use of the same artificial data as explained in Section 5.2.1.

6.2.2 CROSS-VALIDATION DATA

We make use of the same CV datasets described in Section 5.2.2 and summarised in Table 5.1.

6.2.3 REAL-WORLD DATA

We make use of the same RW datasets described in Section 5.2.3 and summarised in Table 5.2.

6.2.4 DATA PRE-PROCESSING

We make use of class-specific and global PCA5 and PCA1 data pre-processing as described in Section 5.2.4 for CV and RW datasets.

6.2.5 METHODS

We compare the MLE, global MLE, MLL, spherical covariance ML-LOO and full covariance ML-LOO estimators as derived in Chapter 4.

6.2.6 KERNEL BANDWIDTH INITIALISATION

In Chapter 5, we compared the performance of conventional density estimators and the MLE rule-of-thumb estimator on a representative set of artificial, CV and RW datasets. We showed that there were several regularities in the relative performance of conventional kernel bandwidth estimators across different tasks and dimensionalities. In particular, we found that the Silverman and MSP estimators consistently performed well across most tasks and dimensions for the artificial, CV and RW datasets investigated and the that performance the MLE rule-of-thumb estimator improved relative to the conventional estimators as dimensionality increased.

Given the consistent performance of the Silverman rule-of-thumb estimator, and the intuitive theoretical motivation that the Silverman estimator optimises the AMISE (assuming a Gaussian reference distribution), we concluded that the Silverman estimator is the preferred conventional density estimator on the datasets investigated in this study. However, it is important to note that this does not necessarily imply that the Silverman estimator will perform optimally as the default bandwidth initialisation procedure, since the bandwidths to which the ML estimators (presented in Chapter 4) converge are dependent on their initialisation, and are thus susceptible to local minima (as illustrated in [17]). Therefore, it is possible that a different initialisation procedure might yield a higher entropy score at initialisation but converges to ML bandwidths that yield a lower entropy score. Given that the MLE rule-of-thumb estimator guarantees that bandwidths are initialised within the range of the optimal ML bandwidths, and is also derived from the ML criteria, we perform a number of tests in Appendix B to compare the performance of the ML estimators when initialised with the Silverman rule-of-thumb and MLE rule-of-thumb estimators. These results show that the MLE rule-of-thumb estimator does in fact frequently yield better performance for the MLE and MLL estimators, and that this improved performance does not hold for the ML-LOO(f) and ML-LOO(s) estimators.

However, since Silverman initialisation gives more consistent performance behaviour across the different ML estimators compared, and does not give a clear advantage to the MLL and MLE estimators, we prefer to use the Silverman estimator as the bandwidth initialisation algorithm for our comparative experiments. We do, however, recognise that bandwidth initialisation has a notable effect on the results of ML estimators, and that other estimators might be even more advantageous than the Silverman and MLE rule-of-thumb estimators considered. It is also possible that no bandwidth initialisation procedure will be optimal for all types of data; therefore this will require a comprehensive study of its own, and is a matter for further research. The focus of this study is to gain a better understanding of the relative performances of the ML estimators considered, and therefore we prefer the consistency of the Silverman estimator for bandwidth initialisation.

For this experiment, the Silverman initial bandwidths were adapted to suit the initialisation of the spherical covariance ML-LOO and full covariance ML-LOO estimators, since the Silverman estimator estimates a unique bandwidth per dimension across all training data points. For the spherical covariance ML-LOO estimator we use the average bandwidth across all dimensions as the initial spherical bandwidth. For the full covariance ML-LOO estimator we initialise the diagonal components of the co-

variance matrix with the respective bandwidths for each dimension; the full covariance ML-LOO estimator is thus initialised with a diagonal covariance matrix.

6.2.7 KERNEL BANDWIDTH REGULARISATION

In practice the optimal ML kernel bandwidth, h_{id}, centred on data point x_i in dimension d, is degenerate under certain circumstances and tends to 0. Pre-processing with PCA reduces the number of such occurrences significantly as explained in Section 5.2.4. As a further measure of regularisation we make use of the theoretical lower bound of the optimal bandwidth for the MLE criterion discussed in Section 4.6. The lower bound states that the minimum optimal bandwidth, h_{id}^2, is equal to the squared Euclidean distance to the nearest-neighbour of x_i. In practice it often happens that two samples have the same values in a certain dimension, thus making the nearest-neighbour distance 0. We thus set the lower bound of a bandwidth in each dimension to the nearest data point with a non-zero distance. For all estimators we validate that all bandwidths are above their respective lower bounds, after each iteration of the bandwidth optimisation procedure. The bandwidths that are below their respective lower bounds after an iteration are thus replaced by the corresponding lower bound value.

6.2.8 SELECTING THE OPTIMAL NUMBER OF TRAINING ITERATIONS

For artificial data, we train the estimators for 10 iterations on each training set, and select the optimal training model as the model with the lowest entropy score across 10 iterations. Similarly for CV datasets, the estimators are trained for 10 iterations on each training fold. The likelihood scores across all folds are combined for each of the 10 models produced by an iteration, and the model with the lowest entropy is selected as the model with optimal CV entropy. For RW datasets, we calculate the CV entropy score (as for the CV data), on the training set for the 10 iterations. The optimal number of iterations is then selected as the CV model with the lowest entropy scores. We then train the estimators for 10 iterations on the full training set, and calculate the entropy score of the test set for each of the 10 models. We select the optimal test entropy, by selecting the test entropy of the model with the optimal number of iterations (selected by CV entropy on the training set).

6.2.9 PERFORMANCE EVALUATION

We use the same performance evaluation methodology as described in Section 5.2.7, with the additional consideration of selecting the optimal training model to test on as described in Section 6.2.8.

6.2.10 STATISTICAL DIFFERENCE TESTING

Each entropy result provided in the results section is calculated as the negative of the mean of the log-likelihood scores (thus the sample entropy as given in Eq. 4.3.3), where the likelihood scores are obtained by substituting the test samples into the optimal training model.

For each class, the test likelihood scores of the estimator with lowest entropy are compared to the remaining four estimators with statistical difference tests. We thus test whether the estimator with the lowest entropy performs significantly better than the remaining estimators with a 5% significance level.

Since the statistical difference test that may be employed is determined by the assumptions of the difference between the likelihoods of the optimal estimator, x, and the likelihoods of the other estimator, y, we first test the assumptions of the distribution $z = x - y$. As explained in Section 2.6, the null hypothesis of the paired t-test is that z is a normally distributed random variable with zero mean and unknown variance. We therefore first need to test for normality before this test can be employed. The null hypothesis of the Kolmogorov-Smirnov test for normality is that z is normally distributed with 0 mean and 1 standard deviation. We therefore first normalise z, prior to testing z for normality with the KS test. If the decision for the null hypothesis is true, the variable z is regarded as normally distributed, and the paired t-test is performed. Since the paired t-test tests whether z has zero mean, and assumes normality, the null hypothesis will be true if the mean is sufficiently close to zero, thus indicating that x and y are not significantly different at the 5% significance level.

If the Kolmogorov-Smirnov test indicates that z is not normally distributed, the Wilcoxon signed-rank test is considered next. The null hypothesis of the Wilcoxon signed-rank test is that z has a zero median, and that z has a continuous distribution that is symmetric around the median. We therefore need to test whether z is symmetric around the median before the Wilcoxon signed-rank test is employed. We make use of the g-statistic hypothesis test for symmetry to determine whether z is symmetric at the 5% significance level. If the null-hypothesis that z is symmetric is accepted by the g-statistic hypothesis test, the Wilcoxon signed-rank test is employed at the 5%

significance level. The null hypothesis of the Wilcoxon rank-sum is that two continuous random variables x and y have equal medians, and it does thus not assume that the difference between x and y has a normal or symmetric distribution. We therefore employ this test if z did not pass the Kolmogorov-Smirnov test for normality and did not pass the g-statistic hypothesis tests for symmetry.

We empirically observed that the distributions of the likelihood scores, x and y, in many cases do not follow a normal distribution, but rather a distribution similar to the log-normal distribution. We therefore considered two alternatives, where $z = x - y$ and where $z = \log(x) - \log(y)$. We found for $z = x - y$, that z did not pass the Kolmogorov-Smirnov test for normality (at a significance level of 5%) for any of the 300 tests that were employed, and that z passed the test for symmetry in only one instance, thus leading to 299 rank-sum tests for the CSPEC, PCA1 data. (Note that there are 75 classes for the five RW datasets in total, and for each of these classes four difference tests are performed.)For $z = \log(x) - \log(y)$, we found that z passed the test for normality for 98 of the 300 tests, and passed the test for symmetry for only seven of the remaining 202 tests. In summary, the paired t-test was performed 98 times, the Wilcoxon signed-rank test seven times, and the Wilcoxon rank-sum test was employed for the remaining 195 tests.

Another motivation for making use of the difference $z = \log(x) - \log(y)$ as the test statistic, is that if we take the negative of the mean of z, then we obtain the sample entropy. The paired t-test will thus measure whether the means of $\log(x)$ and $\log(y)$ are sufficiently close at a significance level of 5%. The paired t-test thus measures if the entropy scores are statistically different or similar. The Wilcoxon signed-rank and rank-sum tests, however, measure if the median values of $\log(x)$ and $\log(y)$ are sufficiently close at a significance level of 5%. These tests thus do not measure directly if the entropy scores are similar, but rather if the median values of the log-likelihood scores $\log(x)$ and $\log(y)$ have similar medians. The median value of log-likelihood scores is not affected by outliers as much as the mean value of log-likelihood scores. Typically outliers are caused by test data points that fall within regions of very low density, thus resulting in very small likelihood values; outliers may also be caused where a test data point falls very close to or directly on a kernel with very small bandwidth centred on a training data point. The Wilcoxon tests thus give a comparison of estimator performance when the effects of such outliers is reduced. This alternative comparison of performance is also interesting, since entropy is very sensitive to outliers; e.g. if we have a set of likelihood measures, and all likelihoods are unremarkable except for a single likelihood value that is very low, then the entropy score will be very low, since

the sample entropy is similar to calculating the product of all likelihood scores.

Theoretically, different tests can be employed between the estimator with the lowest entropy score and the remaining four estimators per class, where the type of test is determined by the assumptions we make regarding z as explained above. For comparative purposes, however, we select only one type of difference test per class, so that the four outcomes of the tests per class can be compared. Therefore, for each class we select the appropriate difference test by selecting the test that will suit the assumptions of all four test statistics, z, for the class. If we assign the paired t-test a value of 1, the Wilcoxon signed-rank test a value of 2 and the Wilcoxon rank-sum test a value of 3, then for each class we take the highest value assigned to any of the four difference tests that were selected based on the assumptions, i.e. if for a class we determine that the following tests can be performed [1 2 3 3], then we select the Wilcoxon rank-sum test as the test to perform for all four tests; if for a class we determine that the tests [1 2 1 2] can be performed, we perform the Wilcoxon signed-rank test for the four tests. Thus only if the tests [1 1 1 1] can be performed, do we employ the paired t-test.

In Section 6.4.3 we report on the outcome of the statistical difference tests for each class and indicate which type of test was selected for each class.

6.3 EXPERIMENTAL DESIGN

In this section we describe the experimental design of the simulation studies performed on artificial, CV and RW data in Experiments 1-3, respectively.

6.3.1 EXPERIMENT 1

We compare the performance of the ML estimators listed in Section 6.2.5 on the artificial datasets as described in Section 6.2.1. The artificial datasets are not pre-processed, and kernel bandwidth initialisation and regularisation are performed as described in Sections 6.2.6 and 6.2.7. The optimal number of training iterations for each estimator is selected as described in Section 6.2.8 and the performances of the estimators are estimated as described in Section 6.2.9.

6.3.2 EXPERIMENT 2

We compare the performance of the ML estimators listed in Section 6.2.5 on the CV datasets as described in Section 6.2.2. The CV datasets are pre-processed as described in Section 6.2.4 and kernel bandwidth initialisation and regularisation are performed

as described in Sections 6.2.6 and 6.2.7. The optimal number of training iterations for each estimator is selected as described in Section 6.2.8 and the performances of the estimators are estimated as described in Section 6.2.9.

6.3.3 EXPERIMENT 3

We compare the performance of the ML estimators listed in Section 6.2.5 on the RW datasets as described in Section 6.2.3. The RW datasets are pre-processed as described in Section 6.2.4 and kernel bandwidth initialisation and regularisation are performed as described in Sections 6.2.6 and 6.2.7. The optimal number of training iterations for each estimator is selected as described in Section 6.2.8 and the performances of the estimators are estimated as described in Section 6.2.9.For each set of entropy results, we also report on the statistical difference tests as described in section 6.2.10.

We also show the CV and test entropy results of the ML estimators over the number of training iterations to illustrate the convergence of the estimators on different tasks.

6.4 RESULTS

The results of Experiments 1-3 are presented in this section. We denote the MLE estimator as MLE, the global MLE estimator as MLE(g), the MLL estimator as MLL, the spherical covariance ML-LOO estimator as ML-LOO(s) and the full covariance estimator as ML-LOO(f).

6.4.1 EXPERIMENT 1

Figures 6.1-6.12 show the comparative entropy results of the ML estimators listed in Section 6.2.5 on Artificial Sets 1-7 for dimensionalities ranging from 1 to 10 and 10 to 100.

Artificial Set 1 contains data sets sampled from the standard MVN distribution. Figure 6.1 shows that all estimators perform competitively on the one- to three-dimensional datasets. The MLL and MLE estimators underperform for dimensionalities higher than 3, while the remaining estimators remain competitive. Given the performance difference between the MLE and MLE(g) estimator, it is clear that the degradation in performance of the MLE and MLL algorithms for dimensionalities above 3 is due to the large number of free parameters these estimators need to estimate. The MLE estimator only requires the estimation of D bandwidths (i.e. a bandwidth for each dimension), while the MLE and MLL estimators required ND bandwidths (i.e. a

bandwidth for each training data point in each dimension). We have shown in Section 3.3 that the number of data points required to achieve a specific MSE increases exponentially as the dimensionality increases. Since the number of training data points remains constant across dimensionalities in this experiment, data sparsity increases significantly. These results thus indicate that the performance of the MLE and MLL estimators is more adversely affected by the curse of dimensionality than the MLE(g), ML-LOO(s) and ML-LOO(f) estimators, since these estimators require the estimation of a larger number of bandwidths.

Figure 6.2 shows that the effect of the curse of dimensionality on the MLE and MLL estimators increases as dimensionality is further increased, and that the performance of the MLL estimator is impaired even more drastically than the MLE estimator on the 40- to 100-dimensional datasets. We also see that the MLE(g) and ML-LOO(s) have almost identical results across all dimensions. This is to be expected, since the theoretical performance of the estimators will be identical if the data is spherically distributed. We also see that the ML-LOO(f) estimator outperforms the ML-LOO(s) and MLE(g) estimators on the 90- and 100-dimensional datasets. Since the data is uncorrelated, the ML-LOO(f) estimator should asymptotically have the same performance as the ML-LOO(s) and MLE(g) estimators. However, for finite datasets, the MLL-LOO(f) estimator has additional co-variance parameters to estimate, impacting its performance.

In *Artificial Set 2* we introduce scale variations by increasing the standard deviations of the features systematically. Figure 6.3 shows that the ML-LOO(s) estimator underperforms for dimensionalities of 3 and higher. This is to be expected, since the ML-LOO(s) estimator estimates the same bandwidth for each dimension, and thus does not model scale variations. The remaining estimators perform competitively over all dimensions.

Figure 6.4 shows that the relative performance of the ML-LOO(s) decreases even further as dimensionality increases. Again we observe that the relative performance of the MLL and MLE estimators decreases as dimensionality increases (similar to Figure 6.2). (Note that the scale change between Figure 6.2 and Figure 6.4 makes the relative difference between the ML-LOO(f) and MLL estimator seem smaller; the absolute difference in entropy, however, remains approximately 0.1.) The remaining estimators perform competitively over all higher dimensions, similar to Artificial Set 1.

In *Artificial Set 3* we introduce correlation to the MVN distribution with features of different scales. Figure 6.5 shows that the estimators perform competitively on dimensions 1 to 3, and that the relative performance of the ML-LOO(f) estimator

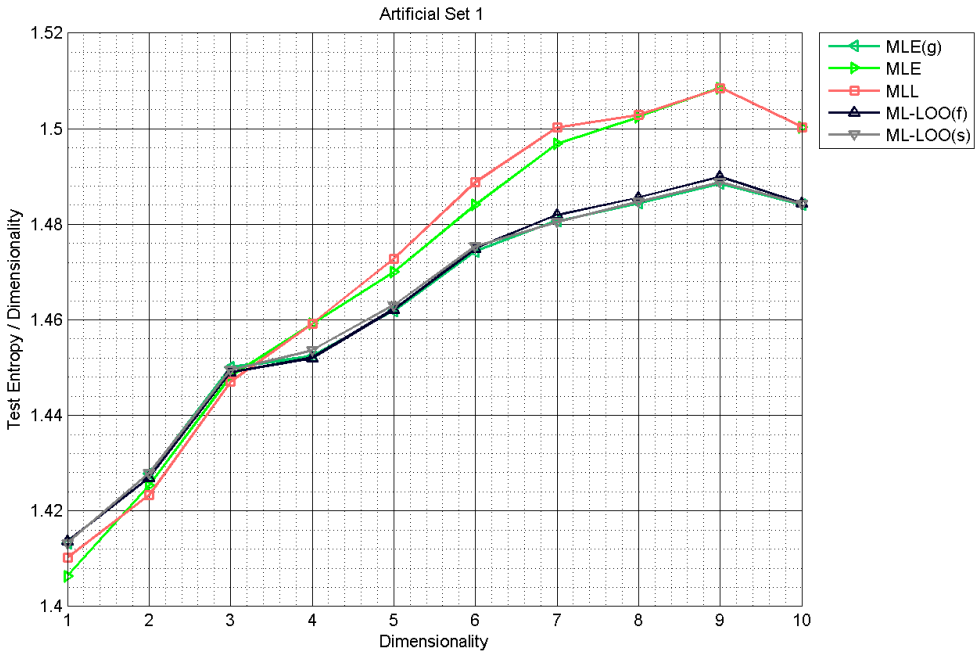

Figure 6.1: *Test entropy results for Artificial Set 1 (D 1-10)*

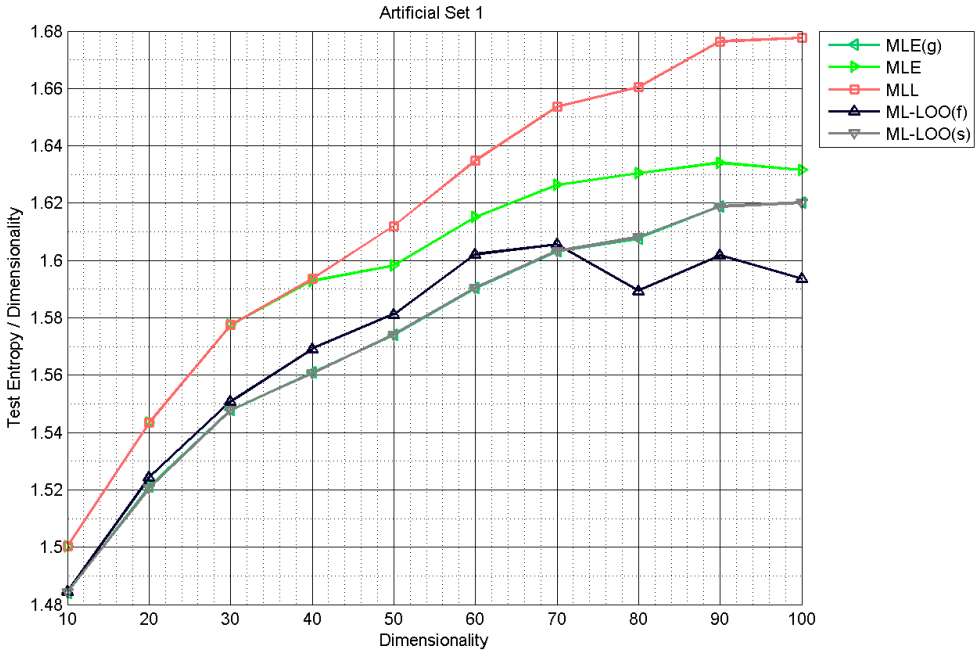

Figure 6.2: *Test entropy results for Artificial Set 1 (D 10-100)*

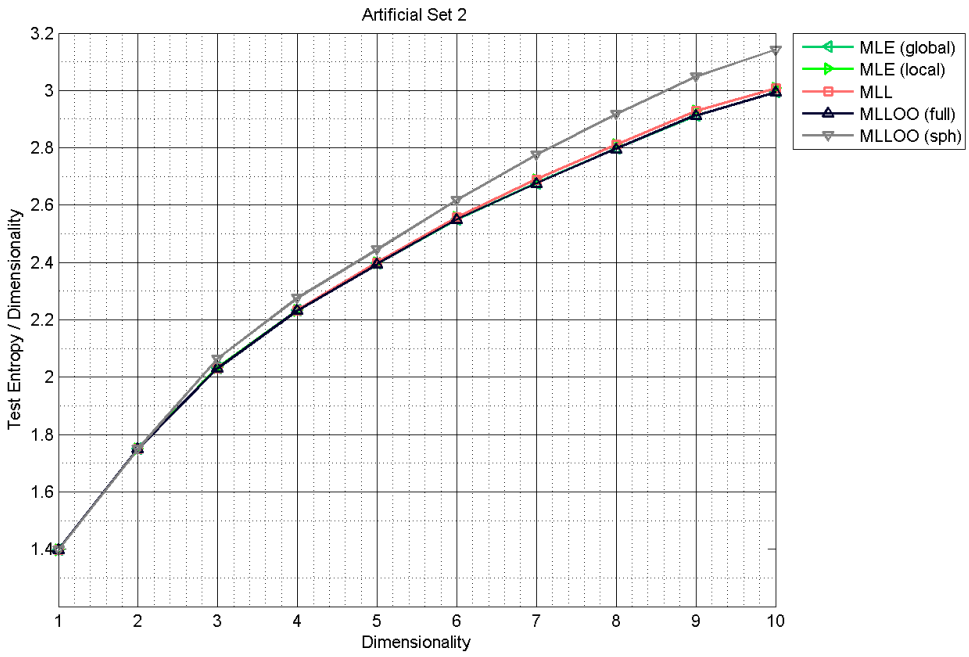

Figure 6.3: *Test entropy results for Artificial Set 2 (D 1-10)*

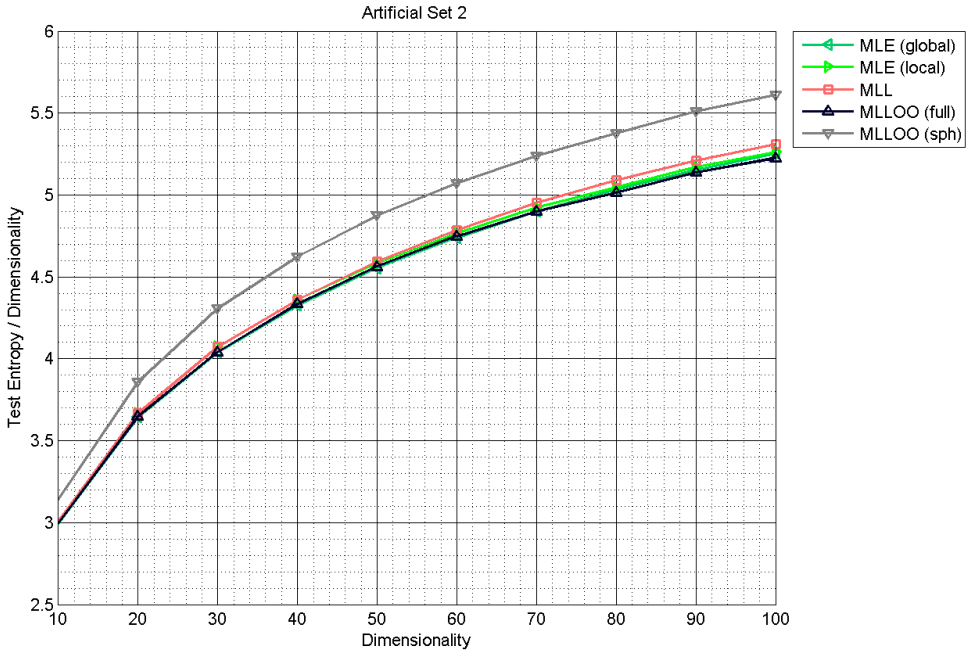

Figure 6.4: *Test entropy results for Artificial Set 2 (D 10-100)*

improves as dimensionality increases and is clearly optimal for dimensions of 5 and higher. This is to be expected, since the ML-LOO(f) estimator estimates an identical full covariance bandwidth matrix for all data points, and thus models correlation. The remaining estimators do not estimate correlation, and perform similarly across all dimensions.

Figure 6.6 shows that the relative increase in performance of the ML-LOO(f) estimator increases even more as dimensionality increases, and that the performance of the remaining estimators are similar over all dimensions.

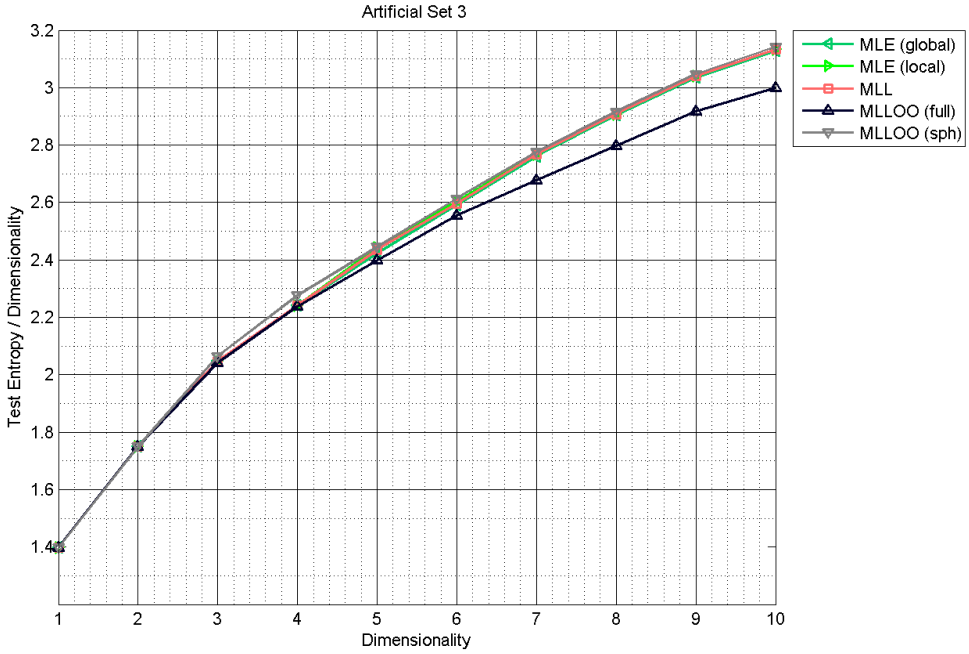

Figure 6.5: *Test entropy results for Artificial Set 3 (D 1-10)*

In *Artificial Set 4* we sample datasets from two standard MVN distributions to simulate bimodal data. Figure 6.7 shows that the ML-LOO(f), ML-LOO(s) and MLE(g) estimators perform competitively across all dimensions (with the exception of the MLL and MLE estimators that perform optimally on the one-dimensional data). The relative performance of the MLE and MLL estimators decrease as dimensionality increases. It is expected that the ML-LOO(f), ML-LOO(s) and MLE(g) estimators will perform similarly, since the data contains no scale variations and no correlation.

Figure 6.8 shows that the relative performance of the ML-LOO(f), ML-LOO(s) and MLE(g) estimators decreases as dimensionality increases, and that the ML-LOO(f) estimator underperforms for all dimensions. Since the ML-LOO(s) and MLE(g) estimators do not estimate correlation, the underperformance of the ML-LOO(f) estimator might

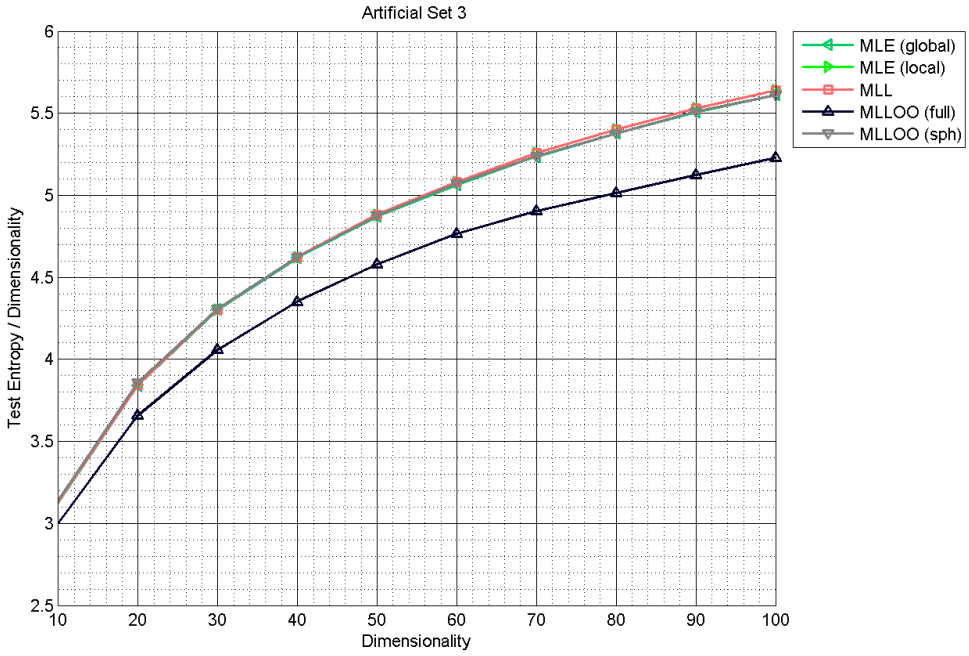

Figure 6.6: *Test entropy results for Artificial Set 3 (D 10-100)*

be caused by the estimation of correlation coefficients in the full-covariance bandwidth matrix that should theoretically be zero (since the data has no correlation). The effect of the estimated correlation coefficients thus seems to increase as data becomes sparser with an increase in dimensionality. We note that for the unimodal MVN data in Figure 6.2, the same behaviour was not observed, thus the estimation of false correlation coefficients might be caused by the bimodality of the data. Surprisingly, the relative performance of the MLL and MLE estimators improves as dimensionality is increased. This shows that these estimators estimate the bandwidths of bimodal data more accurately than the remaining estimators in high dimensions.

In *Artificial Set 5* we introduce intra feature scale variations to the bimodal data as explained in the experimental design. Figure 6.9 shows that the ML-LOO(s) estimator underperforms for dimensionalities of 5 and higher, as is to be expected, since this estimator does not model scale variations between features. The MLL and MLE estimators perform optimally over all dimensions. This is expected since these estimators model scale variations within and between features, by estimating a unique bandwidth for each kernel. The MLE(g) and ML-LOO(s) estimators model scale variations between features (since the bandwidths differ across features), but do not model scale variations within features (since bandwidths do not differ across data points).

Figure 6.10 shows that the relative performance of the MLE and MLL estimators

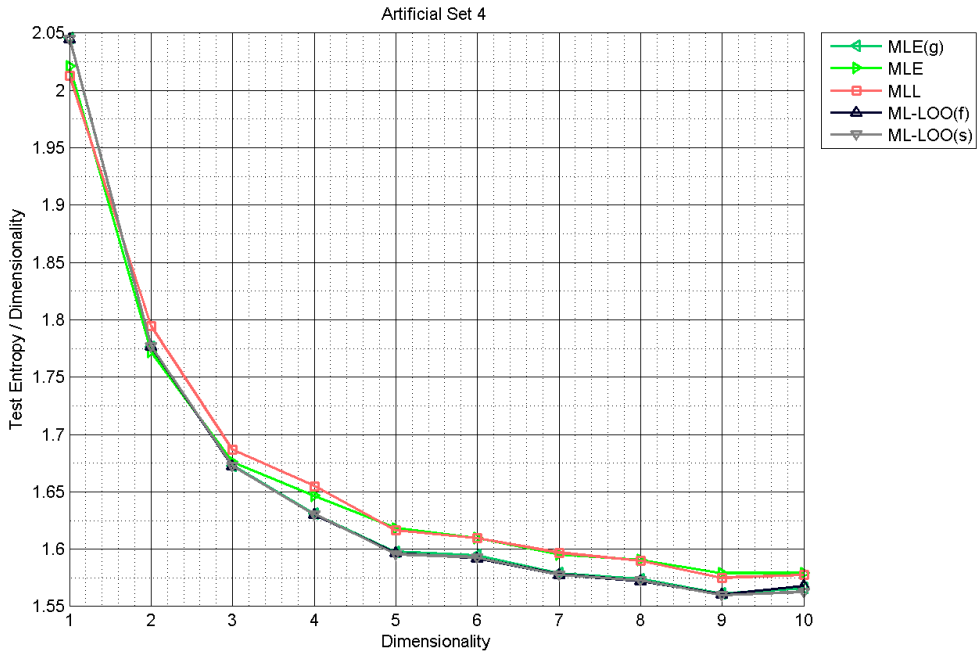

Figure 6.7: *Test entropy results for Artificial Set 4 (D 1-10)*

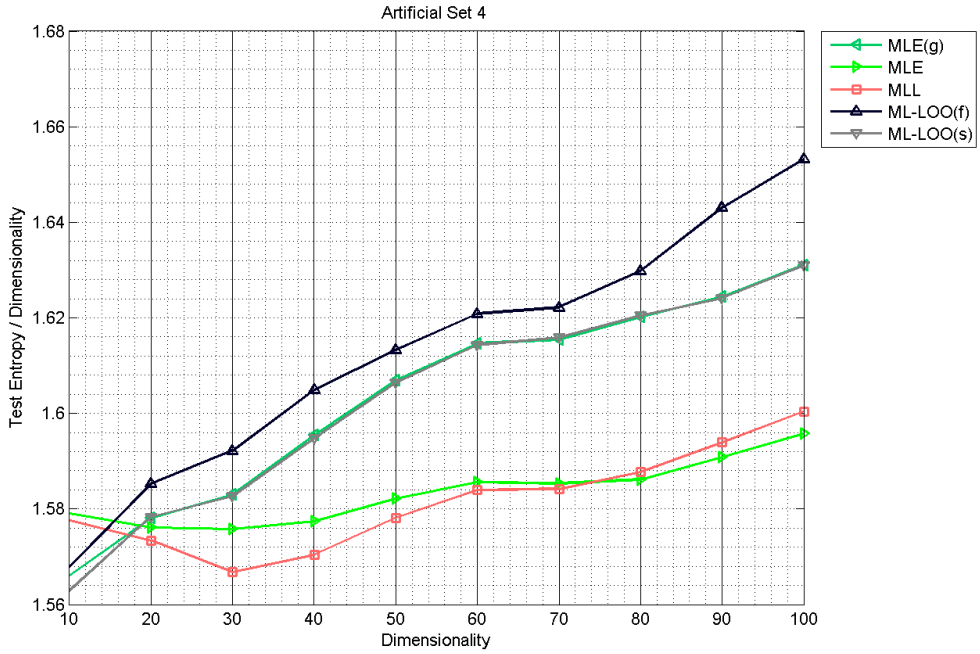

Figure 6.8: *Test entropy results for Artificial Set 4 (D 10-100)*

improves even more as dimensionality increases, and that the remaining estimators underperform over all dimensions.

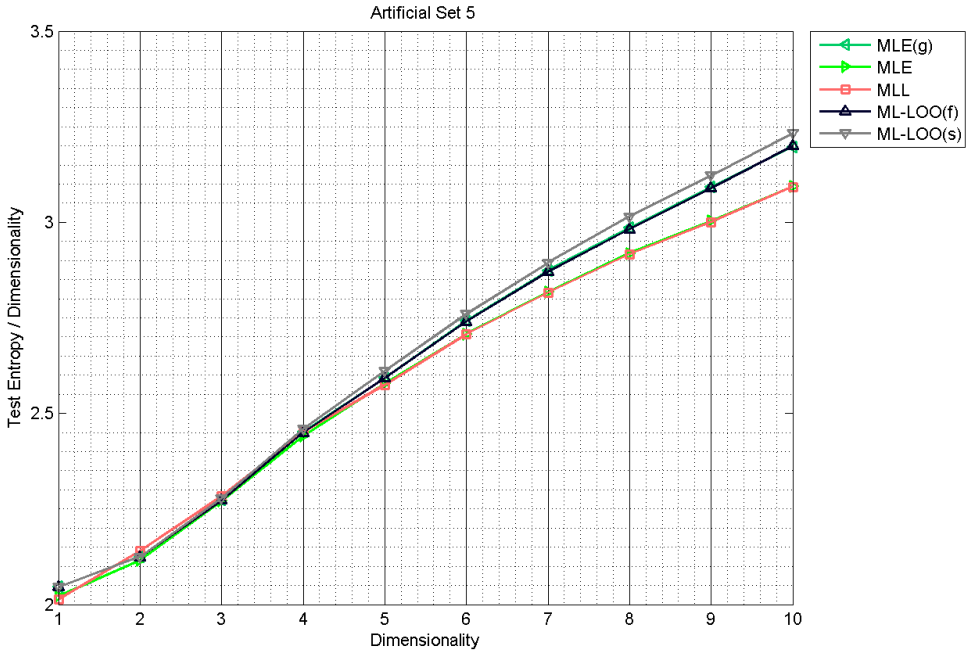

Figure 6.9: *Test entropy results for Artificial Set 5 (D 1-10)*

In *Artificial Set 6* we introduce correlation between the features of the bimodal data with intra-feature scale variations. Figure 6.11 shows that the estimators perform competitively on dimensions 1 to 3, and that the ML-LOO(f) estimator performs optimally on the remaining dimensions. This shows that the introduction of correlation between features gives the ML-LOO(f) estimator a significant improvement over the other estimators. Note that even though the data is bimodal, the covariance matrices of the modes are identical. Thus the bimodality of the data does not influence the relative performance of the ML-LOO(f) estimator.

In Figure 6.12 we see that the relative performance of the ML-LOO(f) estimator improves even more as dimensionality increases, and that this estimator performs optimally across all the dimensions. We also observe that the remaining estimators perform similarly for dimensions of 10-30, and that the MLL estimator experiences a relative performance degradation compared to the MLE, MLE(g) and ML-LOO(s) estimators for dimensionalities of 40 to 100. The MLE(g) and ML-LOO(s) estimator performances are similar. This is to be expected since the scale variations are within features, and not between features. The average scale per dimension is thus similar for this data, although the scale in each dimension varies depending on the mode from which the

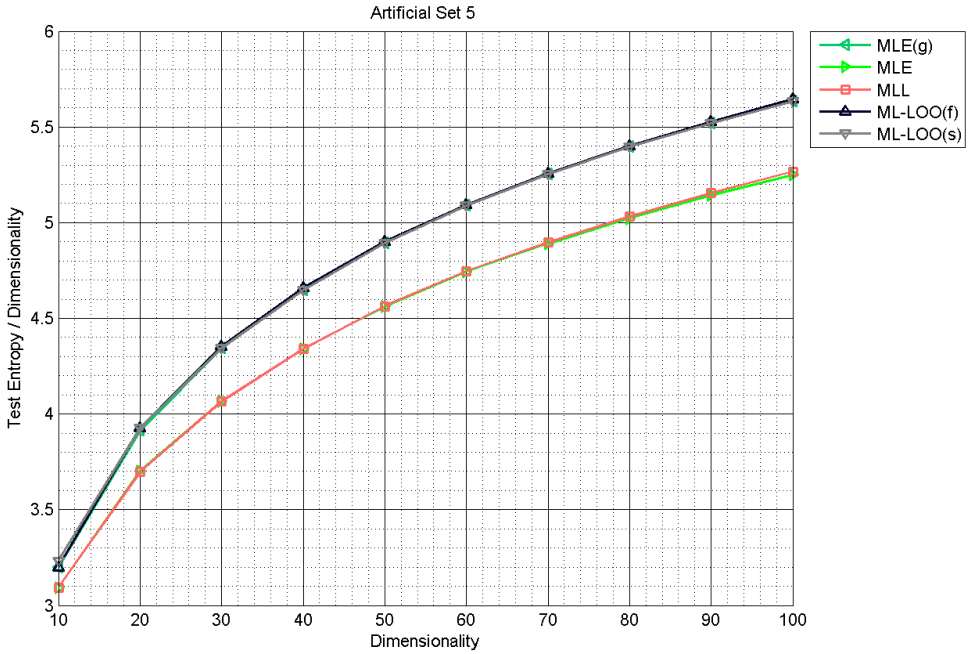

Figure 6.10: *Test entropy results for Artificial Set 5 (D 10-100)*

data is sampled. The MLE(g) estimator underperforms against the ML-LOO(s) and MLE(g) estimators, which is surprising since this estimator is capable of modelling intra-feature scale variations.

In *Artificial Set 7* the covariance matrices of the two MVN Gaussian modes are not identical, since the two modes point in opposite directions in the first dimension. Figure 6.13 shows that tall estimators perform competitively for dimensions 1 to 4, and the ML-LOO(f) estimator performs optimally for dimensions 5 to 10. The remaining estimators perform similarly on all dimensions.

In Figure 6.14 we see that the MLL and MLE estimators underperform for dimensions 40 to 100. The performance of the ML-LOO(f) estimator remains close to optimal, but seems to degrade as dimensionality increases (e.g. for the 90- and 100-dimensional data, the ML-LOO(s) and MLE(g) estimators perform better than the ML-LOO(f) estimator). This indicates that the ML-LOO(s) is not influenced too much by the difference in covariance matrices between the two modes. This might be due to our parameterisation of the covariance matrices, since the data in the two modes only differ in one dimension, which makes the estimation of the same covariance for both modes more realistic.

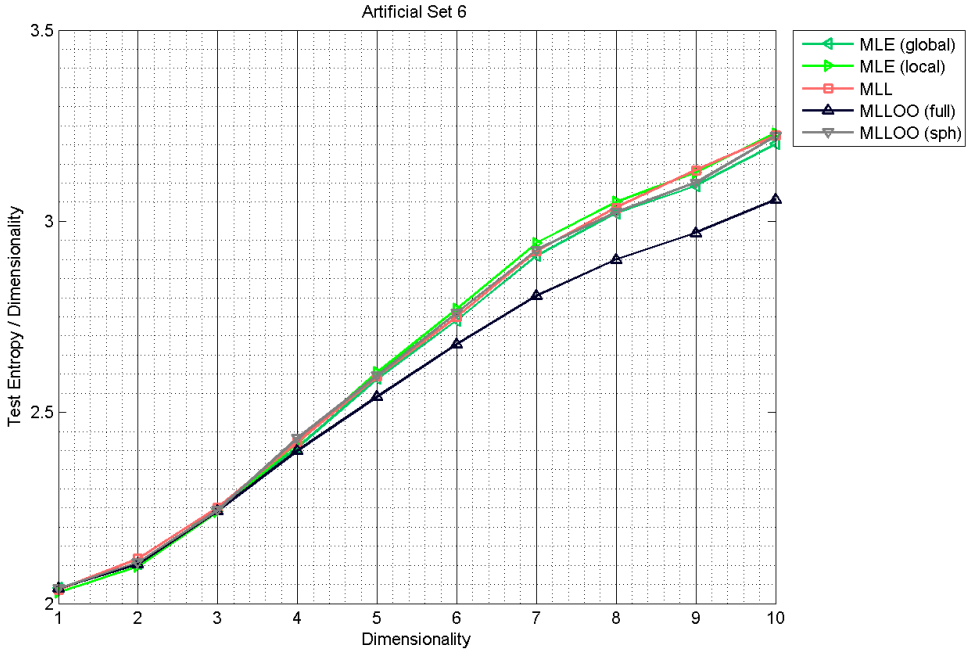

Figure 6.11: *Test entropy results for Artificial Set 6 (D 1-10)*

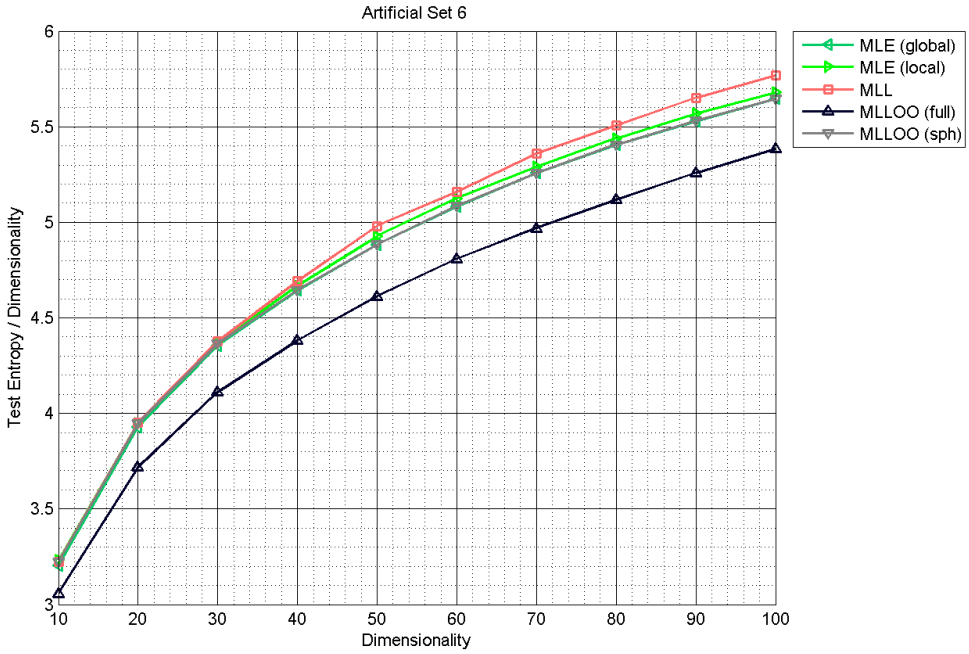

Figure 6.12: *Test entropy results for Artificial Set 6 (D 10-100)*

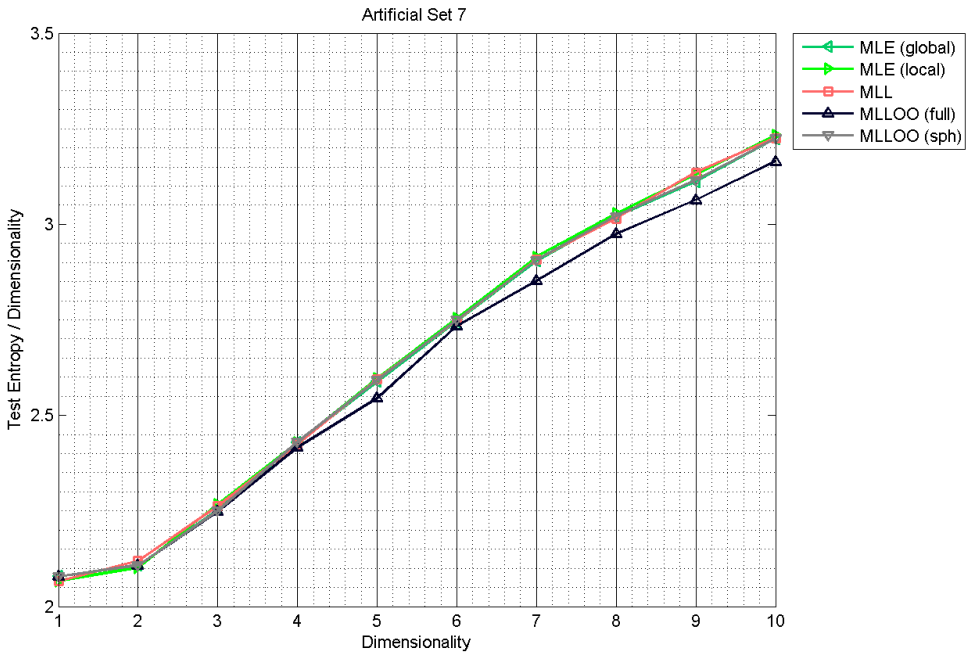

Figure 6.13: *Test entropy results for Artificial Set 7 (D 1-10)*

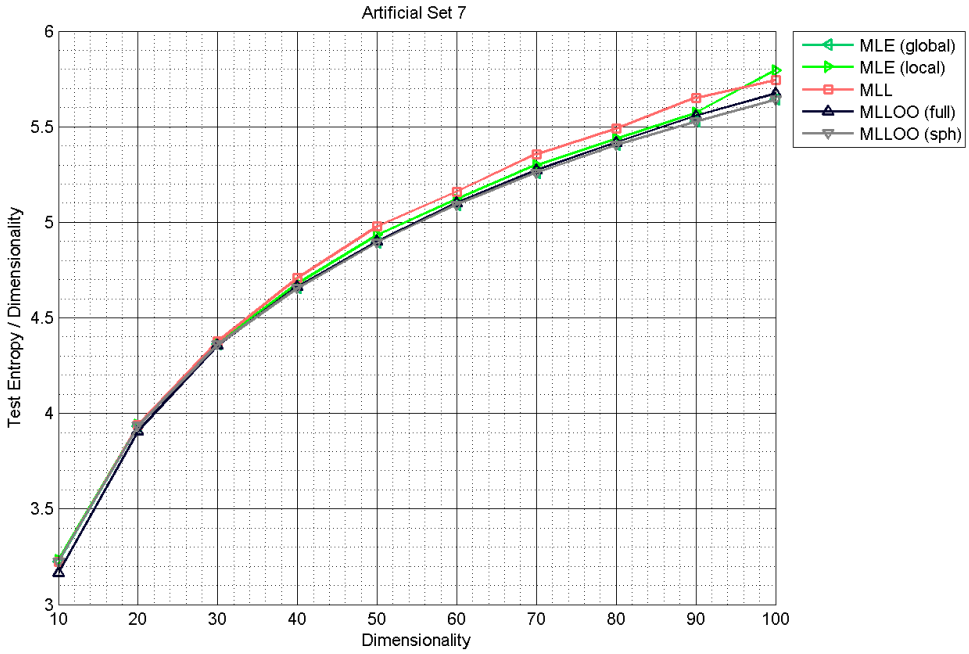

Figure 6.14: *Test entropy results for Artificial Set 7 (D 10-100)*

6.4.2 EXPERIMENT 2

Tables 6.1 and 6.2 show the comparative CV entropy results of the ML estimators on the CV datasets for class-specific PCA5 and PCA1 pre-processing respectively. Note that where the dimensionality of a dataset is less than 5, PCA5 will only decorrelate the data and the dimensionality thus remains the same. The dataset name is indicated with "DS", the class number with "C", the dimensionality of the class after pre-processing with "K", and we abbreviate the dataset names with two-letter acronyms. We make use of colour to represent the relative performance of the estimators visually, where green indicates the lowest entropy for a specific class and red the highest entropy for a specific class.

Table 6.1: CV entropy results (class-specific PCA5)

DS	C	K	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
OF	1	2	1.4815	1.4641	1.4653	1.4781	1.4946
BS	1	4	5.4476	-1.0283	2.1900	5.4503	5.4660
	2	4	-26.1479	-26.1479	-26.5435	-28.7634	5.3657
	3	4	5.4418	-1.2970	2.2335	5.4446	5.4624
IR	1	4	5.5414	5.5205	5.5474	5.5789	5.6942
	2	4	4.8158	4.8840	4.8840	4.8442	4.9101
	3	4	5.0435	5.0798	5.0798	5.0776	5.1759
DB	1	5	7.2291	7.1022	7.1175	7.2268	7.2467
	2	5	7.4277	7.4778	7.4778	7.3686	7.4364
HS	1	5	8.1620	8.1620	8.1620	8.1270	8.1702
	2	5	8.1965	8.2260	8.3019	7.9948	8.2015
VC	1	5	8.2779	8.2684	8.2923	8.2903	8.3712
	2	5	8.2926	8.2375	8.2715	8.2881	8.2997
	3	5	7.1625	6.9422	7.0612	7.6507	7.2193
	4	5	7.7114	8.0964	8.0964	7.7848	7.8292
WF	1	5	8.2643	8.2850	8.2850	8.2658	8.2655
	2	5	8.2132	8.2345	8.2345	8.2145	8.2115
	3	5	8.1898	8.2147	8.2147	8.1909	8.1885
IS	1	5	9.6743	9.6604	9.9790	9.7254	9.6488
	2	5	8.3689	8.5333	8.5752	8.3399	8.4064

The *Old Faithful* dataset results in Table 6.1 show that MLE and MLL estimators perform competitively on the two-dimensional dataset, and that the ML-LOO(s) estimator underperforms. The underperformance of the ML-LOO(s) estimator indicates that there are scale variations between features, and the superior performance of the

Table 6.2: CV entropy results (class-specific PCA1)

DS	C	K	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
OF	1	2	1.4815	1.4641	1.4653	1.4781	1.4946
BS	1	4	5.4476	-1.0283	2.1900	5.4503	5.4660
	2	3	4.8336	4.8649	4.8649	4.8546	4.8720
	3	4	5.4418	-1.2970	2.2335	5.4446	5.4624
IR	1	4	5.5414	5.5205	5.5474	5.5789	5.6942
	2	4	4.8158	4.8840	4.8840	4.8442	4.9101
	3	4	5.0435	5.0798	5.0798	5.0776	5.1759
DB	1	8	9.8422	9.9005	9.9455	9.6644	9.9346
	2	8	10.4833	10.4459	10.5278	10.3307	10.4336
HS	1	13	18.1319	18.5913	18.5913	14.3833	18.3462
	2	13	17.5738	17.7804	17.7804	10.8952	17.7250
VC	1	9	10.5104	10.8288	10.9117	10.3832	11.2043
	2	8	9.8030	10.0864	10.2151	9.6618	10.4235
	3	8	8.9059	8.7243	8.8984	9.3708	9.2334
	4	11	11.1508	11.3137	11.6883	11.6020	12.0935
WF	1	21	26.7635	27.1857	27.1857	26.7922	27.4237
	2	21	29.2129	29.6012	29.6012	29.2531	29.5631
	3	21	29.1510	29.5804	29.5804	29.1949	29.5093
IS	1	33	42.4526	41.6154	46.9883	46.9883	45.7913
	2	12	13.5870	12.9432	14.2051	13.7731	14.2113

MLE and MLL estimators indicate that there might be scale variations within features.

The *Balance Scale* dataset results in Table 6.1 show that the MLE estimator performs optimally with a large margin on classes 1 and 3, while the ML-LOO(f) estimator performs optimally on class 3. The optimal performance of the MLE estimator on classes 1 and 3 is consistent with the results presented in Appendix B; we refer the reader to Section B.2 for a more detailed analysis of estimator performance on these two classes.

The PCA1 results in Table 6.2 show similar relative performance behaviour on the PCA1 data. Note that only the dimensionality of class 2 has changed, and that the excellent performance of the MLE(g), MLE, MLL and ML-LOO(f) increases when the dimensionality is reduced from four to three dimensions. This indicates that the dimension that was dropped with the PCA1 transformation might have contained useful information.

The *Iris* dataset results in Tables 6.1 and 6.2 show that the PCA5 and PCA1 transformations reduce the dataset to the same intrinsic dimensionality; the results are thus identical. We observe in these tables that the MLE(g) estimator performs optimally on classes 2 and 3, and close to optimally on class 3. The ML-LOO(s) estimator again underperforms on all classes because of the variation in eigenvalues between features.

The *Diabetes* dataset results in Table 6.1 show that the MLE and MLL estimators perform competitively on class 1 of the five-dimensional dataset, and that the ML-LOO(f) estimator performs optimally on class 2, while the MLE and MLL estimators underperform on class 2. The PCA1 results in Table 6.2 show that the ML-LOO(f) estimator performs optimally on the eight-dimensional PCA1 dataset, and that the MLL estimator underperforms. The increase in dimensionality thus increased the relative performance of the ML-LOO(f) estimator on both classes, while the relative MLL estimator experienced a relative performance degradation. This is consistent with the MLL behaviour on Artificial Set 1 and Artificial Set 3.

The *Heart(Statlog)* dataset results in Table 6.1 show that the ML-LOO(f) estimator performs optimally on both classes of the five-dimensional dataset, while the ML-LOO(s) estimator underperforms on class 1 and the MLL estimator underperforms on class 2. The PCA1 results in Table 6.2 show that the performance of the ML-LOO(f) estimator remains optimal on the eight-dimensional PCA1 datasets, and that the MLE and MLL estimators experiences a severe performance degradation with the increase in dimensionality.

The *Vehicle* dataset results in Table 6.1 show that the MLE estimator performs

optimally on classes 1-3 of the five-dimensional dataset, while the MLE(g) estimator performs optimally on class 4. Interestingly, the MLE and MLL estimators underperform on class 4. The PCA1 results in table 6.2 show that the ML-LOO(f) estimator performs optimally on classes 1 and 2, while the MLE estimator performs optimally on class 3, and the MLE(g) estimator on class 4. We also observe that the MLE(g) estimator has the best general performance over all classes.

The *Waveform* dataset results in Table 6.1 show that the MLE(g) and ML-LOO(s) estimators perform competitively on all classes of the five-dimensional dataset. The good performance of the ML-LOO(s) estimator indicates that the eigenvalues of the five features are similar, thus causing the transformed data to be close to spherical. The PCA1 results in Table 6.2 show, that the MLE(g) estimator performs optimally on all classes of the higher-dimensional data, and that the ML-LOO(s) estimator experiences a severe degradation in relative performance. This implies that the additional features that were added for the high-dimensional data, added local structure to the data that is not suited to spherical bandwidth matrices as used by the ML-LOO(s) estimator. The results in Table 6.1 also show that the MLE and MLL estimators underperform on the five-dimensional data on all classes. These estimators also underperform on classes 2 and 3 of the higher-dimensional PCA1 data. Finally, the results in Table 6.1 and Table 6.2 show that the ML-LOO(f) estimator experiences an increase in relative performance as dimensionality increases, and performs competitively on the higher-dimensional PCA1 data.

The *Ionosphere* dataset results in Tables 6.1 and 6.2 show that all estimators, except for the MLL estimator, perform competitively on the five-dimensional dataset, and that the MLL estimator also underperforms on the higher-dimensional PCA1 data. We also observe in Table 6.2 that the MLE estimator performs optimally on both classes of the PCA1 dataset. We note as well that the entropy scores of class 1 on the PCA1 dataset are much higher than the entropy scores of class 2 on the PCA dataset owing to the large difference in dimensionality (33 as opposed to 12).

In summary, we have observed in this experiment that the MLE and MLE(g) estimators performed well on most classes of the five-dimensional PCA5 datasets, and the MLE(g) performed well on most classes of the higher-dimensional PCA1 datasets, followed by the ML-LOO(f) and MLE estimators. The ML-LOO(f) estimator experienced an improvement in relative performance as dimensionality increased, while the MLL and ML-LOO(s) estimators experienced a degradation in performance as dimensionality increased.

We report on the results of the global PCA5 and PCA1 datasets in Appendix C,

where we make general comments on the differences between the class-specific and global PCA results.

6.4.3 EXPERIMENT 3

The *Letter* dataset results in Tables 6.3 and 6.4 show that the ML-LOO(f) estimator performs optimally on most classes for both the five-dimensional PCA5 data, and the higher-dimensional PCA1 data. The ML-LOO(s) estimator, on the other hand, underperforms on most classes for the PCA5 and PCA1 data. The MLL estimator performs moderately for most classes of the PCA5 data, and experiences relative performance degradation for the PCA1 data. Conversely, the MLE(g) estimator underperforms on a number of classes for the PCA5 data, and experiences a relative performance improvement on the PCA1 data. The MLE estimator performs well on a number of classes for the PCA5 data, and similar to the MLL estimator, experiences a relative degradation in performance as dimensionality increases for the PCA1 data.

The relative degradation in performance of the MLL and MLE estimators when the dimensionality of the data is increased from PCA5 to PCA1 might be explained by the significant increase in the number of parameters to estimate when the dimensionality increases. For example, the dimensionality of class 1 in the Letter dataset increases from 5 to 15, while the number of samples in class 1 is fixed at 789. The number of parameters to estimate (which we summarised in Table 4.1) for the MLL estimator thus increases from ND=789x5 for the PCA5 dataset to ND=789x15 for the PCA1 dataset. The ratio between the number of samples and number of parameters will thus decrease from 0.2 samples/parameter to 0.067 samples/parameter, which leads to an increase in data sparsity that might explain the relative degradation in performance.

Conversely, the relative improvement in performance of the ML-LOO(f) and MLE(g) estimators might be explained by the smaller number of parameters to estimate when compared to the MLL and MLE estimators. For the Letter dataset class 1, the ML-LOO(f) estimator will estimate $D(D+1)/2$ parameters per class. For the PCA5 dataset, the ML-LOO(f) estimator will thus estimate 5(5+1)/2=15 parameters and for the PCA1 dataset 15(15+1)/2=120 parameters. The ratio between the number of samples and number of parameters will thus decrease from 52.5 samples/parameter to 6.575 samples/parameter. Even in the 15-dimensional case, the ML-LOO(f) estimator has 6.575/0.2=32.875 times more samples/parameter than the MLE and MLL estimators for the five-dimensional case. The MLE(g) estimator needs to estimate even fewer parameters than the ML-LOO(f) estimator (only D parameters), and therefore the number of samples per parameter will be even higher than that of the ML-LOO(f)

estimator; for PCA5 data the ratio is 789/5=157.8 samples/parameter and for PCA1 data the ratio is 789/15=52.6 samples/parameter.

This example illustrates that the number of samples per parameter to be estimated should be considered when selecting an estimator, and that estimators with fewer parameters might lead to better generalisation performance when the number of samples/dimension ratio is very small.

Table 6.3: Letter test entropy results (class-specific PCA5)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	5	789	6.1653	6.1786	6.1750	5.9772	6.2620
2	5	766	7.4753	7.2150	7.2107	7.2271	7.6006
3	5	736	7.1287	7.1089	7.1023	6.9025	7.1505
4	5	805	6.7478	6.3395	6.5235	6.5860	6.7890
5	5	768	6.5361	6.4266	6.4462	6.1686	6.5595
6	5	775	6.8708	6.7795	6.7891	6.7111	6.8970
7	5	773	7.0607	6.9871	6.9752	6.8964	7.0529
8	5	734	6.5832	6.4309	6.4479	6.0982	6.5792
9	5	755	5.8674	5.2918	5.4175	5.6007	5.9317
10	5	747	6.5710	6.4170	6.4262	6.4027	6.6224
11	5	739	6.7653	6.6544	6.6964	6.5889	6.7564
12	5	761	5.8378	5.6657	5.7381	5.5734	5.8204
13	5	792	6.0048	6.0412	6.0817	5.8205	6.1440
14	5	783	6.2698	6.0737	5.9840	6.1940	6.4741
15	5	753	7.6773	7.4707	7.4034	7.5396	7.6481
16	5	803	7.0580	7.0263	7.1152	6.9694	7.1261
17	5	783	6.8312	6.9166	6.7962	6.6505	6.8761
18	5	758	7.0294	7.1174	6.9854	6.8324	7.0414
19	5	748	6.4988	6.5811	6.5529	6.2064	6.5550
20	5	796	6.4601	6.4457	6.4933	6.2711	6.4866
21	5	813	6.2496	6.1510	6.2548	6.0624	6.2960
22	5	764	6.6892	6.7701	6.8066	6.5264	6.7054
23	5	752	7.0167	6.9532	6.9839	6.7140	7.3462
24	5	787	6.5997	6.4873	6.5736	6.4897	6.8772
25	5	786	6.1810	6.2319	6.1728	6.0200	6.2323
26	5	734	6.9033	6.6380	6.6154	6.7288	7.0168

We provide the statistical difference test results for the *Letter* dataset in Tables 6.5 and 6.6. We denote the type of difference test as "TE", where the paired t-test is represented by "1", the Wilcoxon signed-rank test by "2" and the Wilcoxon rank-sum

Table 6.4: Letter test entropy results (class-specific PCA1)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	15	789	9.2873	10.2996	10.5922	7.9763	10.1448
2	15	766	11.8521	12.4504	13.2300	10.6214	12.5788
3	15	736	10.9114	10.9152	11.4798	10.2541	11.8211
4	14	805	10.6806	10.8480	11.2551	9.8986	11.1240
5	15	768	9.0948	10.0307	10.6545	7.4805	9.8392
6	13	775	9.3735	9.8056	10.0067	8.2964	10.0040
7	15	773	11.6837	12.4585	12.7124	10.6560	12.3793
8	14	734	10.6712	10.6509	10.6427	8.9762	11.4651
9	13	755	8.7703	7.4315	7.4505	7.6537	9.1916
10	13	747	8.0048	8.2247	8.5730	6.9398	8.5478
11	15	739	10.1285	10.7803	10.6981	9.1266	11.0117
12	12	761	6.9545	7.0843	7.2469	5.9676	7.4122
13	14	792	8.3489	8.4041	8.7171	6.7962	9.0203
14	14	783	9.3522	8.7992	9.7257	8.4742	10.1868
15	16	753	12.7908	13.0909	13.9332	11.5703	13.7332
16	15	803	9.9907	11.1816	11.5641	9.4843	10.8242
17	15	783	10.8606	11.8567	12.0359	9.6828	11.8571
18	15	758	10.9078	12.1141	12.4231	9.9914	11.3758
19	15	748	9.0280	9.8751	10.0427	7.5570	9.7640
20	13	796	7.8998	8.1263	8.4534	6.6746	8.7879
21	13	813	8.8809	9.0425	9.2017	7.9219	9.2352
22	14	764	10.0613	10.3505	10.6758	8.7185	10.8547
23	15	752	10.5571	10.9732	11.4986	9.3987	11.8726
24	15	787	11.0415	11.1224	11.6708	9.7728	12.2909
25	14	786	8.7009	9.3734	9.8168	7.4867	9.2290
26	15	734	10.2744	10.0557	10.8011	9.1422	11.2895

test by "3". Note that these results correspond to the entropy results in Tables 6.3 and 6.4. We indicate the estimator with optimal entropy for a specific class as "-", a significant difference at the 5% significance level as 1, and a non-significant difference at the 5% significance level as 0. We note that statistical differences indicate superior performance of the estimator with the lowest entropy score, since for each class the optimal estimator is compared to the other estimators. We also note that wherever we refer to statistically significant, the significance level is 5%.

It is important to note that the non-parametric Wilcoxon signed-rank and rank-sum tests are typically less powerful in determining differences between two paired sets of measurement, compared to the parametric paired t-test. In addition, owing to the fact that we perform an independent test per class, our statistical tests are not sensitive to the regularities that appear across classes. We will show in our results that for most classes in which the Wilcoxon tests are employed, the test statistics are generally not normally distributed at a significance level of 95%; therefore we observe regularities in entropy scores that are not always indicated by the Wilcoxon difference tests, because of the relatively low statistical power of our tests. We will highlight such cases in our discussion of the results.

The *Letter* class-specific PCA5 statistical difference test results show that the ML-LOO(f) estimator performs significantly better than the ML-LOO(s) estimator on six of the 26 classes – this confirms our observation on Table 6.3. The ML-LOO(f) estimator also performs significantly better than the MLE(g), MLE and MLL estimators on two, one and three classes respectively. The estimators perform statistically similarly on the remaining tests. These results confirm our conclusion that the ML-LOO(f) estimator generally performs better than the other estimators, and that the ML-LOO(s) estimator generally underperforms for this dataset. We observe that the Wilcoxon rank-sum test was used for all classes, except class 3, which indicates that the test data was not normally distributed or symmetric for at least one estimator for each of the 25 classes.

The *Letter* class-specific PCA1 statistical difference test results show that the relative performance of the ML-LOO(f) estimator improved compared to the PCA5 results, since there are more classes where statistical differences were found between the ML-LOO(f) estimator and the other estimators. From these results it is clear that the ML-LOO(f) estimator performs optimally on all classes 3, 17 and 23, and performs statistically similarly compared to the other estimators on the remaining classes. These results confirm our conclusion drawn from Table 6.4, that the ML-LOO(f) estimator generally performs optimally, the ML-LOO(s) estimator generally performs suboptimally and that the MLE and MLL estimators experience a relative degradation in

Table 6.5: Letter statistical difference test results (class-specific PCA5)

C	K	NC	TE	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	5	789	3	0	0	0	-	1
2	5	766	3	0	0	-	0	0
3	5	736	1	1	0	1	-	1
4	5	805	3	0	-	0	0	0
5	5	768	3	0	0	0	-	1
6	5	775	3	0	0	0	-	0
7	5	773	3	0	0	0	-	0
8	5	734	3	0	0	0	-	1
9	5	755	3	0	-	0	0	0
10	5	747	3	0	0	0	-	0
11	5	739	3	0	0	0	-	0
12	5	761	3	0	0	0	-	0
13	5	792	3	0	0	0	-	0
14	5	783	3	0	0	-	0	0
15	5	753	3	0	0	-	0	0
16	5	803	3	0	0	0	-	0
17	5	783	3	0	0	0	-	0
18	5	758	3	0	0	0	-	0
19	5	748	3	1	1	0	-	1
20	5	796	3	0	0	1	-	0
21	5	813	3	0	0	0	-	0
22	5	764	3	0	0	1	-	0
23	5	752	3	0	0	0	-	1
24	5	787	3	0	-	0	0	0
25	5	786	3	0	0	0	-	0
26	5	734	3	0	0	-	0	0

performance compared to the PCA5 data. We observe that the Wilcoxon rank-sum test was performed on all classes, except class 5 where the Wilcoxon signed-rank test was performed. The test data for class 5 was thus symmetric around the median, but not normally distributed.

Table 6.6: Letter statistical difference test results (class-specific PCA1)

C	K	NC	TE	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	15	789	3	1	0	0	-	0
2	15	766	3	1	0	1	-	1
3	15	736	3	1	1	1	-	1
4	14	805	3	1	1	0	-	1
5	15	768	2	1	0	0	-	0
6	13	775	3	0	0	1	-	0
7	15	773	3	1	1	0	-	1
8	14	734	3	1	0	0	-	1
9	13	755	3	0	-	0	0	1
10	13	747	3	0	0	1	-	1
11	15	739	3	1	1	0	-	1
12	12	761	3	1	0	0	-	1
13	14	792	3	1	0	0	-	1
14	14	783	3	0	0	0	-	1
15	16	753	3	0	0	1	-	1
16	15	803	3	0	1	1	-	1
17	15	783	3	1	1	1	-	1
18	15	758	3	1	1	0	-	1
19	15	748	3	1	0	0	-	0
20	13	796	3	1	1	1	-	1
21	13	813	3	1	0	0	-	1
22	14	764	3	1	0	0	-	0
23	15	752	3	1	1	1	-	1
24	15	787	3	1	0	0	-	0
25	14	786	3	0	1	1	-	1
26	15	734	3	0	0	0	-	1

The *Segmentation* dataset results in Tables 6.7 and 6.8 show that the MLE and MLL estimators perform competitively on most classes for the PCA5 and PCA1 data, with the exception of class 2 for the PCA5 data and classes 2 and 7 of the PCA1 data. The MLE(g) and ML-LOO(s) estimators underperform on most classes for the PCA5 and PCA1 data. The ML-LOO(f) estimator performs optimally on class 2 for the

PCA5 data, and classes 2 and 7 for the PCA1 data – thus for the same classes where the MLE and MLL estimators underperformed. The ML-LOO(f) estimator, however, does not perform competitively on the remaining classes.

Table 6.7: Segmentation test entropy results (class-specific PCA5)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	5	330	8.7886	8.4363	8.4789	8.7740	9.0469
2	5	330	8.0941	7.7080	7.7931	6.7962	7.9038
3	5	330	5.9737	3.4367	3.6945	6.5749	6.0940
4	5	330	11.0808	6.8538	6.7601	11.6064	8.0821
5	5	330	7.6424	4.6877	5.0381	8.0332	7.7044
6	5	330	7.9777	7.6515	7.7691	8.1101	7.8086
7	5	330	4.7858	4.1647	4.3740	4.4453	5.0026

Table 6.8: Segmentation test entropy results (class-specific PCA1)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	13	330	14.7612	14.3396	14.5128	15.2068	16.6483
2	11	330	13.3975	13.5591	15.3803	12.9305	13.8543
3	12	330	10.8880	2.7401	3.5781	10.5969	9.6920
4	11	330	15.4156	7.7225	11.9012	17.5817	13.2768
5	11	330	15.4297	6.7261	7.2938	9.9464	17.4564
6	11	330	11.2927	10.7688	11.0891	11.3063	11.7564
7	10	330	7.1770	6.8176	6.8371	3.1038	7.3919

The statistical difference test results for the PCA5 *Segmentation* dataset in Table 6.9 show that the MLE estimator performs significantly better than the MLE(g), ML-LOO(f) and ML-LOO(s) estimators on class 3, and the MLE(g) and ML-LOO(s) on class 5. The MLL estimator also performs significantly better than the ML-LOO(s) estimator on class 4. The remaining estimators perform statistically similarly (or competitively) for all other results.

The statistical difference test results for the PCA1 *Segmentation* dataset in Table 6.10 show that the MLE performs significantly better than the MLE(g) estimator on three classes and significantly better than the ML-LOO(f) and ML-LOO(s) estimators on two classes. The ML-LOO(f) estimator performed significantly better than all other estimators on class 7, and the MLL estimator performs statistically similarly to the optimal estimators for all classes, except class 7. These results confirm our conclusions

(on Tables 6.7 and 6.8) that the MLE and MLL estimators perform competitively on most classes for the PCA5 and PCA1 data.

Table 6.9: Segmentation statistical difference test results (class-specific PCA5)

C	K	NC	TE	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	5	330	3	0	-	0	0	0
2	5	330	3	0	0	0	-	0
3	5	330	1	1	-	0	1	1
4	5	330	3	0	0	-	0	1
5	5	330	3	1	-	0	0	1
6	5	330	3	0	-	0	0	0
7	5	330	3	0	-	0	0	0

Table 6.10: Segmentation statistical difference test results (class-specific PCA1)

C	K	NC	TE	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	13	330	2	0	-	0	0	0
2	11	330	3	0	0	0	-	0
3	12	330	3	1	-	0	1	0
4	11	330	3	1	-	0	1	1
5	11	330	3	1	-	0	0	1
6	11	330	3	0	-	0	0	0
7	10	330	1	1	1	1	-	1

The *Landsat* results in Tables 6.11 and 6.12 show that the MLE(g) estimator performs competitively on class 5 for both PCA5 and PCA1 data. The MLE estimator performs optimally on five classes of the PCA5 data, experiences relative performance degradation for the PCA1 data and only performs optimally on two classes on the PCA1 data. Similarly, the MLL estimator performs optimally on one class on the PCA5 data, and experiences degradation in performance with the increase in dimensionality of the PCA1 data. Conversely, the ML-LOO(f) estimator performs competitively on only two classes of the PCA5 data, and experiences a relative performance improvement on the PCA1 data on which the ML-LOO(f) estimator performs competitively on five classes.

Similarly to the Letter dataset, we observe degradation in the performance of the MLE and MLL estimators when the dimensionality of the dataset is increased from PCA5 to PCA1, and we observe an improvement in performance of the MLE(g) and ML-LOO(f) estimators. This again might be explained by the relative decrease in the

number of samples/parameter as dimensionality increases, as in the example given for the Letter dataset.

Table 6.11: Landsat test entropy results (class-specific PCA5)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	5	1533	8.3463	8.3396	8.3249	8.3519	8.4456
2	5	703	8.1212	7.8695	7.9616	8.1172	8.2582
3	5	1358	9.4238	9.3575	9.4478	9.4210	9.5558
4	5	626	8.4932	8.4234	8.5772	8.4374	8.8392
5	5	707	9.5215	9.5349	9.5361	9.5220	9.5678
6	5	1508	9.0352	8.9075	8.9776	9.0256	9.2311

Table 6.12: Landsat test entropy results (class-specific PCA1)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	8	1533	10.6246	10.6571	10.6958	10.5527	11.0285
2	10	703	11.4557	11.2188	11.2724	11.3094	12.0536
3	14	1358	16.6442	16.7766	17.1863	16.4641	17.8911
4	13	626	14.6303	14.6583	14.7188	14.6150	15.8079
5	11	707	14.1853	14.6757	14.6641	14.2129	14.8374
6	11	1508	14.2989	14.2268	14.5458	14.2334	15.0980

The statistical difference test results for the *Landsat* PCA5 dataset in Table 6.13 show that all estimators performed statistically similarly, except for the ML-LOO(s) estimator that performed significantly worse on three of the six classes. The optimal performance of the MLE estimator observed in Table 6.11 and the relative degradation in performance observed in Table 6.12 are thus not statistically significant. We also note that although the ML-LOO(s) estimator underperforms on all classes of the PCA5 data, the difference tests only indicate a statistically significant difference for three of the five classes. This shows that the hypothesis test of the Wilcoxon rank-sum test statistic might be overly weak to detect that two samples are significantly different (at a 5% significance level). If the results for all classes were combined, it is likely that this test would indicate that the ML-LOO(s) estimator performs significantly worse than the other estimators.

The PCA1 results in Table 6.14 show that the relative performance of the ML-LOO(s) estimator degrades, and that for the higher-dimensional data the ML-LOO(s)

estimator performs significantly worse on all classes. These results also show that the MLE(g) estimator performs statistically similarly to the optimal estimator for all classes. In addition, they show that the relative degradation in performance of the MLL estimator from PCA5 to PCA1 seen in Tables 6.11 and 6.12 is not statistically significant, since the test results in Table 6.14 show that the MLL estimator performs statistically similarly to the optimal estimators.

Table 6.13: Landsat statistical difference test results (class-specific PCA5)

C	K	NC	TE	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	5	1533	3	0	0	-	0	0
2	5	703	3	0	-	0	0	1
3	5	1358	1	0	-	0	0	0
4	5	626	3	0	-	0	0	1
5	5	707	3	-	0	0	0	0
6	5	1508	3	0	-	0	0	1

Table 6.14: Landsat statistical difference test results (class-specific PCA1)

C	K	NC	TE	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	8	1533	3	0	0	0	-	1
2	10	703	3	0	-	0	0	1
3	14	1358	3	0	0	0	-	1
4	13	626	3	0	0	0	-	1
5	11	707	3	-	0	0	0	1
6	11	1508	3	0	0	0	-	1

The *Optdigits* dataset results in Tables 6.15 and 6.16 show that the ML-LOO(f) estimator performs competitively on five classes of the PCA5 data, and that this performance does not generalise to the PCA1 data, since this estimator does not perform competitively on any of the classes of the PCA1 data. The MLE(g) estimator, on the other hand, performs competitively on only two classes of the PCA5 data, and performs competitively on seven classes of the PCA1 data. This shows that the estimation of correlation affects the ML-LOO(f) estimator adversely as the dimensionality of the classes increases (since the ML-LOO(f) estimator estimates a full covariance bandwidth matrix). The MLE and MLL estimators perform competitively on some classes of the PCA5 data, and experience slight performance degradation on the PCA1 data. The

ML-LOO(s) estimator generally underperforms on the PCA5 data, and performs competitively on more classes of the PCA1 data – this estimator thus experiences a slight improvement in relative performance as dimensionality increases. We note that the MLE and MLL estimators have identical entropy values on the PCA1 class 7 data. We have investigated and confirmed that for this class and other such instances, the estimators do not improve on the initial entropy values when the bandwidths are updated, and since they are initialised with identical bandwidths the initial entropy scores are the same, thus resulting in identical entropy scores. The optimal entropy score is thus the entropy score of the Silverman bandwidth estimate.

As in the Letter and Landsat datasets, we observe degradation in performance of the MLE and MLL estimators when the dimensionality of the dataset is increased from PCA5 to PCA1 for the Optdigits dataset, and we observe an improvement in performance of the MLE(g) estimator. We do not, however, see a relative improvement of the ML-LOO(f) estimator for the Optdigits dataset. This might be because of the significant increase in dimensionality, e.g. the dimensionality increases from 5 to 46 for class 1 between PCA5 and PCA1. The ML-LOO(f) estimator, must estimate $D(D+1)/2$ parameters, which causes the number of parameters to estimate to increase from 5(5+1)/2=15 parameters to 46(46+1)/2=1081 parameters. This causes the number of samples per parameter to decrease from 554/15=36.93 to 554/1081=0.541. As was the case for the Letter and Segmentation datasets, the degradation in performance of these estimators as dimensionality increases might be explained by the relative decrease in the number of samples/parameter as dimensionality increases.

Table 6.15: Optdigits test entropy results (class-specific PCA5)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	5	554	10.4372	10.4789	10.4584	10.4373	10.4915
2	5	571	9.6967	9.6537	9.5757	9.6905	9.7080
3	5	557	10.5627	10.6060	10.5163	10.5040	10.5311
4	5	572	10.3737	10.3141	10.3879	10.5734	10.4704
5	5	568	9.0819	8.7344	8.7181	9.0864	9.2245
6	5	558	10.1294	10.2500	10.2737	10.1951	10.1885
7	5	558	11.0836	10.3956	10.5549	10.9724	10.7495
8	5	566	10.7315	10.6860	10.6933	10.6483	10.7030
9	5	554	10.8419	10.8843	10.8843	10.8239	10.8431
10	5	562	10.2479	10.3508	10.3302	10.2245	10.2248

The statistical difference test results for the *Optdigits* PCA5 dataset in Table 6.17

Table 6.16: Optdigits test entropy results (class-specific PCA1)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	46	554	52.7315	54.9642	54.9642	55.0121	57.2316
2	36	571	37.8930	39.9837	39.9837	40.0183	39.8571
3	41	557	55.1442	60.8972	60.8972	61.1132	55.9001
4	50	572	63.7679	67.9808	67.9808	69.7068	69.8641
5	39	568	45.1030	45.1030	45.1030	45.1332	44.2904
6	46	558	55.7444	58.8748	58.8748	58.9782	59.6404
7	37	558	46.8292	46.7069	46.7069	46.9650	48.8804
8	41	566	52.9752	55.9436	55.9436	55.9758	55.6049
9	50	554	61.2910	63.9965	63.9965	64.2079	65.3601
10	43	562	57.9241	56.2229	56.2229	51.2123	59.1375

show that the performances of all estimators are statistically similar on 8 of the 10 classes. For class 7, the ML-LOO(f) and ML-LOO(s) estimators perform significantly worse than the other estimators, and for class 9 the MLE and MLL estimators perform significantly worse than the other estimators. This confirms our observation in Table 6.15, that the ML-LOO(f) estimator performs competitively.

The PCA1 results in Table 6.18 show that the relative performance of the ML-LOO(s) estimator degrades (similar to the Landsat results) as dimensionality increases, and that the ML-LOO(s) estimator performs significantly worse than the other estimators on five of the 10 classes – this confirms our observation on Tables 6.15 and 6.16. The remaining estimators do not show a statistically significant difference in performance on all classes, except for class 10 where the ML-LOO(f) estimator performs significantly better than all other estimators. These tests show that the relative performance degradation of the ML-LOO(f) estimator in Tables 6.15 and 6.16 is not statistically significant, and also that the optimal performance of the MLE(g) estimator in Table 6.16 is also not statistically significant.

The *Isolet* PCA5 dataset results in Table 6.19 show that no estimator clearly outperforms all other estimators, since each estimator performs optimally on some classes and least optimally on other classes. The PCA1 results in Table 6.20 show that the MLE(g) estimator experiences a relative improvement in performance, and performs optimally on most classes of the higher-dimensional data, while the ML-LOO(s) estimator, conversely, experiences a relative degradation in performance, and underperforms on all classes of the PCA1 data. These results indicate that for the high-dimensional PCA1 data, the variance (i.e. eigenvalues) of the transformed variables differ signifi-

Table 6.17: Optdigits statistical difference test results (class-specific PCA5)

C	K	NC	TE	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	5	554	3	-	0	0	0	0
2	5	571	3	0	0	-	0	0
3	5	557	1	0	0	-	0	0
4	5	572	3	0	-	0	0	0
5	5	568	3	0	0	-	0	0
6	5	558	3	-	0	0	0	0
7	5	558	3	0	-	0	1	1
8	5	566	3	0	0	0	-	0
9	5	554	3	0	1	1	-	0
10	5	562	3	0	0	0	-	0

Table 6.18: Optdigits statistical difference test results (class-specific PCA1)

C	K	NC	TE	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	46	554	3	-	0	0	0	1
2	36	571	3	-	0	0	0	0
3	41	557	3	-	0	0	0	1
4	50	572	3	-	0	0	0	1
5	39	568	3	0	0	0	0	-
6	46	558	3	-	0	0	0	0
7	37	558	3	0	-	0	0	0
8	41	566	3	-	0	0	0	1
9	50	554	3	-	0	0	0	0
10	43	562	3	1	1	1	-	1

cantly (thus leading to poor ML-LOO(s) performance) and that the assumption of no correlation seems to be more suitable (given the difference in performance between the MLE(g) and ML-LOO(f) estimators) for the high-dimensional data.

Again, the good generalisation performance of the MLE(g) estimator in high dimensions might be explained by the small number of parameters that must be estimated. For example, for the PCA1 class 1 data, the MLE(g) needs to estimate only 92 parameters, while the ML-LOO(f) requires the estimation of 92(92+1)/2=4278 parameters and the MLE and MLL estimators require the estimation of ND=300=27600 parameters. Given the small number of training samples per class, these more flexible methods are at a disadvantage.

The statistical difference test results for the *Isolet* PCA5 dataset in Table 6.21 show that all estimators perform statistically similarly except for classes 13 and 21, this confirms our observation on Table 6.19 that no estimator generally performs best. For class 13, the ML-LOO(f) and ML-LOO(s) estimators perform significantly worse than the other estimators, and for class 21 the MLE estimator performs significantly worse than the other estimators.

The PCA1 results in Table 6.22 show that the relative performance of the ML-LOO(s) again degrades as the dimensionality increases. The ML-LOO(s) estimator performs significantly worse than the optimal estimator for 10 of the 26 classes. There is no statistically significant difference in the performance of the remaining estimators and optimal estimators for all classes, except classes 2, 16 and 22. For classes 2 and 22, the MLL, ML-LOO(s) and ML-LOO(f) estimators perform significantly worse than the MLE estimator, and for class 16 the MLE estimator performs significantly worse than the MLE(g) estimator. This shows that the optimal performance of the MLE(g) estimator in Table 6.20 is not statistically significant. Similar to the Landsat dataset, we observe that although the ML-LOO(s) estimator underperforms for all classes of the PCA1 data, the difference tests only indicate a statistically significant difference for 10 of the 26 datasets, again illustrating the weakness of these tests.

We have explained in Section 6.2.8 that the optimal test entropy is calculated by selecting the test entropy of the model with the optimal number of training iterations (selected by CV entropy on the training set). All test entropy results for RW datasets presented thus far, only reported on the optimal test set entropy. We will now investigate the CV and test set entropy of the estimators over the number of training iterations, to gain a better understanding of the convergence performance of the estimators on different tasks. We show the results of only one class per dataset, where this class is selected as the class with the highest intrinsic dimensionality (as determined by

Table 6.19: Isolet test entropy results (class-specific PCA5)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	5	300	15.8607	15.9339	15.9339	15.8813	15.8468
2	5	300	15.2731	15.2619	15.2655	15.1603	15.2114
3	5	300	16.2714	16.2951	16.2951	16.3190	16.2675
4	5	300	15.8201	15.7429	15.7984	15.8697	15.8393
5	5	300	15.9924	16.0375	16.0375	15.9802	16.0121
6	5	300	15.3304	15.2806	15.2870	15.2903	15.4209
7	5	300	15.6160	15.5452	15.5395	15.5965	15.5696
8	5	300	15.7104	15.7602	15.7629	15.6860	15.7867
9	5	300	16.9600	17.2859	17.2859	17.1408	16.9469
10	5	300	15.1778	15.2037	15.2040	15.1900	15.3320
11	5	300	15.7678	15.8327	15.8025	15.8551	15.7736
12	5	300	17.5882	16.5873	16.6023	17.5936	17.5082
13	5	300	14.9260	14.9641	14.9474	14.9642	15.0078
14	5	300	15.0635	15.0635	15.2451	15.0641	15.2452
15	5	300	15.1259	15.0920	15.1016	15.6394	15.2775
16	5	300	15.0720	15.0435	15.1094	15.0096	15.0043
17	5	300	15.8266	15.8046	15.8225	15.8819	15.8094
18	5	300	16.3976	16.3282	16.3282	16.4231	16.4296
19	5	300	16.5864	15.9832	16.0932	16.0988	16.3513
20	5	300	15.4371	15.4574	15.4568	15.3718	15.4746
21	5	300	14.8951	15.0450	14.9417	14.9061	14.9033
22	5	300	16.3253	16.2548	16.3105	16.3106	16.3329
23	5	300	14.9877	15.2028	15.0208	14.9294	15.0812
24	5	300	15.6869	15.6942	15.7648	15.6179	15.7433
25	5	300	15.8849	15.8584	15.8855	15.9412	15.9006
26	5	300	16.4605	16.5053	16.5738	16.5097	16.4618

Table 6.20: Isolet test entropy results (class-specific PCA1)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	92	300	189.0759	191.2731	190.8883	190.8883	205.5452
2	107	300	211.6703	210.8217	212.5131	212.5131	227.0795
3	98	300	203.3099	202.9658	202.9658	202.9675	218.5153
4	101	300	205.9877	209.6424	207.7975	207.8165	220.4039
5	103	300	213.3857	215.8229	218.8905	218.9028	230.3118
6	100	300	199.3268	199.7004	200.6295	200.6515	219.3737
7	90	300	186.8359	189.5903	188.2592	188.2732	198.8510
8	88	300	180.7441	182.4256	181.7845	182.0967	195.6949
9	108	300	230.2376	236.3707	226.6773	226.6732	243.0001
10	85	300	182.6589	184.5528	184.5528	184.6028	193.5564
11	89	300	189.1564	190.9196	190.9196	190.9604	199.9672
12	118	300	241.4164	252.6089	249.7367	250.0967	267.3455
13	123	300	244.3422	248.1532	249.8309	249.8319	269.9968
14	119	300	239.3669	239.3669	261.6026	241.8353	264.3237
15	103	300	209.7407	215.6002	213.0124	213.1697	232.1384
16	104	300	206.8094	208.4278	207.3090	207.3090	221.1164
17	108	300	222.6612	227.7650	227.7650	227.7496	244.1215
18	106	300	217.0238	224.1280	221.0936	221.2660	235.8874
19	96	300	199.3421	201.4060	203.1794	203.7217	219.1281
20	100	300	203.0767	202.6541	202.6541	202.6541	217.6525
21	121	300	239.3333	241.4055	244.0612	244.1695	260.1166
22	113	300	228.0789	226.9123	230.1929	230.1947	248.3391
23	104	300	207.1714	208.6037	209.0199	209.0204	220.9262
24	92	300	194.4274	199.8320	197.9024	197.9059	209.3233
25	109	300	218.3860	219.9983	220.8091	221.0781	239.4411
26	100	300	206.8940	206.6910	206.6910	206.7172	224.4596

Table 6.21: Isolet statistical difference test results (class-specific PCA5)

C	K	NC	TE	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	5	300	3	0	0	0	0	-
2	5	300	3	0	0	0	-	0
3	5	300	1	0	0	0	0	-
4	5	300	3	0	-	0	0	0
5	5	300	3	0	0	0	-	0
6	5	300	3	0	-	0	0	0
7	5	300	3	0	0	-	0	0
8	5	300	3	0	0	0	-	0
9	5	300	3	0	0	0	0	-
10	5	300	3	-	0	0	0	0
11	5	300	3	-	0	0	0	0
12	5	300	3	0	-	0	0	0
13	5	300	3	-	0	0	1	1
14	5	300	3	-	0	0	0	0
15	5	300	3	0	-	0	0	0
16	5	300	3	0	0	0	0	-
17	5	300	3	0	-	0	0	0
18	5	300	3	0	-	0	0	0
19	5	300	3	0	-	0	0	0
20	5	300	3	0	0	0	-	0
21	5	300	3	-	1	0	0	0
22	5	300	3	0	-	0	0	0
23	5	300	3	0	0	0	-	0
24	5	300	3	0	0	0	-	0
25	5	300	3	0	-	0	0	0
26	5	300	3	-	0	0	0	0

Table 6.22: Isolet statistical difference test results (class-specific PCA1)

C	K	NC	TE	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	92	300	3	-	0	0	0	0
2	107	300	1	0	-	1	1	1
3	98	300	3	0	-	0	0	0
4	101	300	3	-	0	0	0	1
5	103	300	3	-	0	0	0	0
6	100	300	1	-	0	0	0	1
7	90	300	3	-	0	0	0	0
8	88	300	3	-	0	0	0	0
9	108	300	3	0	0	0	-	0
10	85	300	3	-	0	0	0	0
11	89	300	3	-	0	0	0	0
12	118	300	3	-	0	0	0	1
13	123	300	3	-	0	0	0	0
14	119	300	3	-	0	0	0	0
15	103	300	3	-	0	0	0	1
16	104	300	1	-	1	0	0	1
17	108	300	3	-	0	0	0	1
18	106	300	3	-	0	0	0	0
19	96	300	3	-	0	0	0	1
20	100	300	3	0	-	0	0	0
21	121	300	3	-	0	0	0	0
22	113	300	1	0	-	1	1	1
23	104	300	3	-	0	0	0	0
24	92	300	3	-	0	0	0	0
25	109	300	3	-	0	0	0	1
26	100	300	3	0	-	0	0	0

the CSPEC PCA1 transformation). We summarise the classes with the highest intrinsic dimensionalities in Table 6.23. We denote the class number as "C_i", the number of training samples for class C_i as "Ntr_i", the number of test samples for class C as "Nte_i", the original dimensionality of class C_i as "D_i" and the intrinsic dimensionality of class C_i as "K_i".

Table 6.23: RW dataset summary for selected classes

Dataset	$\mathbf{C}_i$	$\mathbf{Ntr}_i$	$\mathbf{Nte}_i$	$\mathbf{D}_i$	$\mathbf{K}_i$
Letter	15	608	145	16	16
Segmentation	1	300	30	18	13
Landsat	3	961	397	36	14
Optdigits	4	389	183	62	50
Isolet	13	240	59	617	123

We provide only the class-specific PCA5 and PCA1 results in this chapter, and provide the results of the corresponding classes for global PCA5 and PCA1 in Appendix C. Note that the entropy results for CV are calculated by combining the likelihood scores of each of the samples in the training set; CV entropy is thus calculated for NC_{tr} samples. The test entropy results are calculated by combining the likelihood scores of each of the test data points; test entropy is thus calculated for NC_{te} samples. Table 6.23 shows that NC_{tr} is larger than NC_{te} for all of the RW datasets considered, therefore the CV entropy scores are on average larger than the test entropy scores in the results presented.

The class-specific CV and test entropy scores of class 15 of the PCA5 Letter dataset, for 10 training iterations are shown for each estimator in Figure 6.15. We observe that the MLL and MLE estimators converge to the optimal entropy score within two to three training iterations, and that the entropy score diverges as training is continued. These estimators thus have the advantage that they converge very quickly compared to the other estimators, but are also susceptible to overtraining. Since the CV and test entropy reach the optimal entropy at approximately the same number of iterations, the optimal CV entropy can be used to determine the optimal number of training iterations, and consequently overtraining can be prevented by stopping training when the CV entropy starts to increase. We also observe that the MLL estimator experiences more severe performance degradation when the optimal number of training iterations is exceeded. Thus MLE might suffer less from overtraining, since the MLE estimator has an extra denominator in the bandwidth update equation that reduces overfitting (as explained in Section 4.5.1).

The remaining estimators converge more slowly to the optimal entropy, and are not so susceptible to overtraining, since their performance does not degrade as severely as that of MLL and MLE estimators if training continues after the optimal entropy has been reached. The overtraining of the MLL and MLE estimators might be caused by the unique bandwidth matrix that is estimated for each kernel, as opposed to an identical bandwidth matrix per kernel for the other estimators.

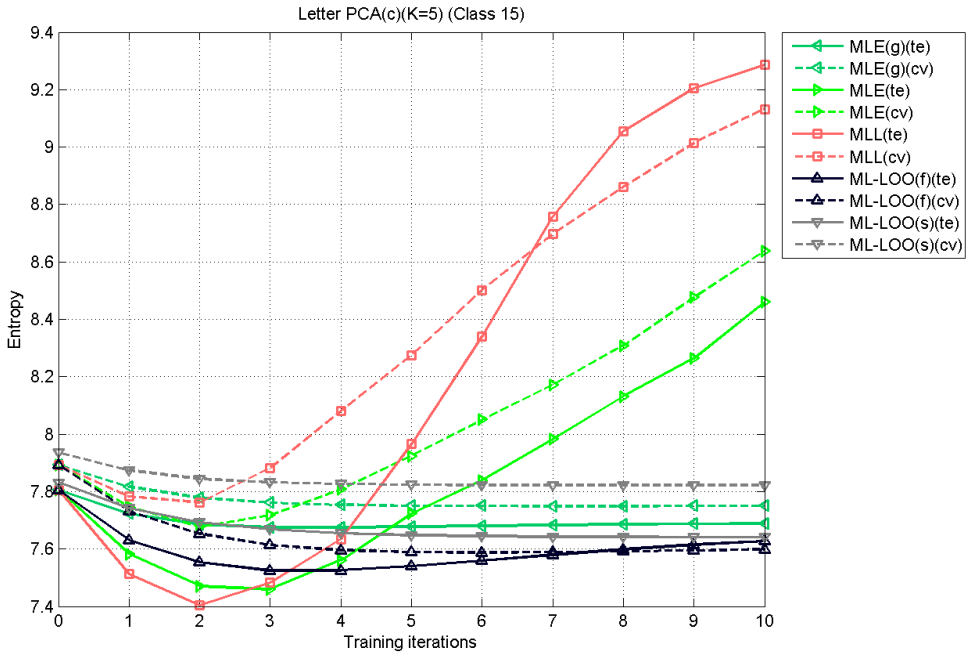

Figure 6.15: *Letter entropy results for training iterations(Class 15, CSPEC, PCA5)*

Figure 6.16 shows the class-specific entropy results for class 15 of the 16-dimensional PCA1 *Letter* dataset. We observe behaviour similar to that of the PCA5 dataset, except that the optimal entropy scores of the MLE and MLL estimators are not lower than those of the MLE(g) and ML-LOO(f) estimators. We also observe that the effect of overtraining on the MLL and MLE estimators seems to be more adverse compared to the lower-dimensional PCA5 dataset.

The class-specific entropy results of class 1 of the PCA5 *Segmentation* dataset in Figure 6.17 shows that the MLL and MLE estimators converge to the optimal entropy in only one iteration, and that they suffer severe performance degradation as training is continued. The entropy scores of the MLE(g) estimator increases relative to the initial entropy score, in this case the Silverman estimator thus gives a better entropy score. Similarly, the test entropy score of the ML-LOO(f) estimator increases relative to the initial Silverman entropy score; the CV entropy scores, however, reach the optimal

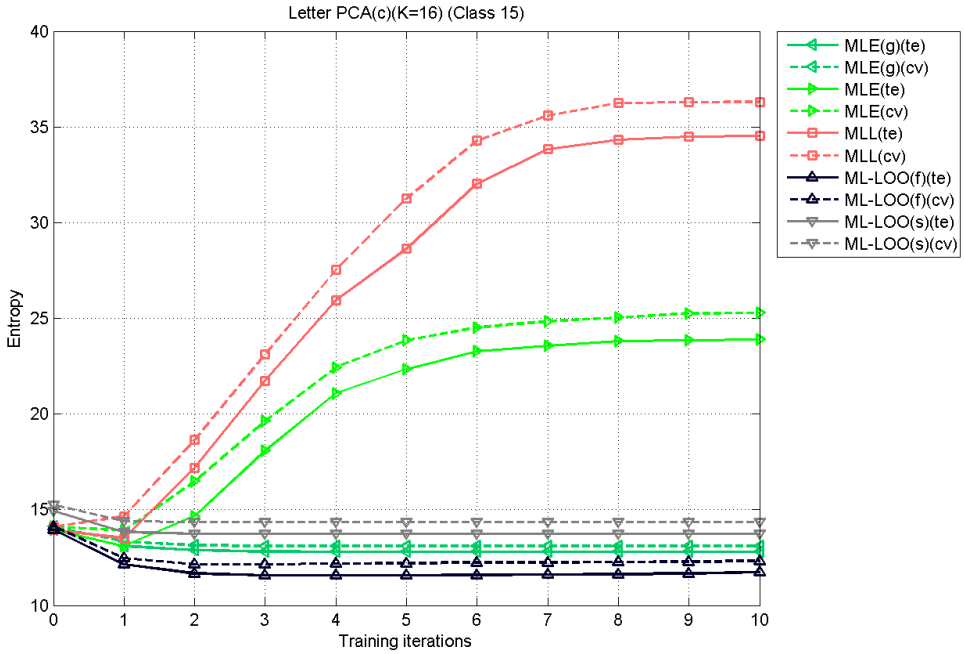

Figure 6.16: *Letter entropy results for training iterations(Class 15, CSPEC,PCA1)*

entropy score after one iteration.

Figure 6.18 shows the class-specific entropy results of class 1 of the 13-dimensional PCA1 *Segmentation* dataset. We again observe that the MLL and MLE estimators converge to the optimal entropy after one training iteration, and that the entropy increases as training is continued. The remaining estimators, again, converge more slowly to the optimal entropy, and are not so susceptible to overtraining, since their performance does not degrade as severely as that of the MLL and MLE estimators if training continues after the optimal entropy has been reached.

The class-specific entropy results of class 3 of the PCA5 *Landsat* dataset in Figure 6.19 shows that the MLL and MLE estimators converge to the optimal entropy in only one iteration and that the remaining estimators converge more slowly to the optimal entropy.

Figure 6.20 shows the class-specific entropy results for class 3 of the 14-dimensional PCA1 *Landsat* dataset. We observe that the MLE estimator converges to the optimal entropy in only one iteration, and the entropy of the MLL estimator increases monotonically relative to the initial Silverman entropy score. Again, the remaining estimators converge more slowly to the optimal entropy.

The class-specific entropy results of class 4 of the PCA5 *Optdigits* dataset in Figure 6.21 show that the MLL and MLE estimators converge to the optimal entropy in only

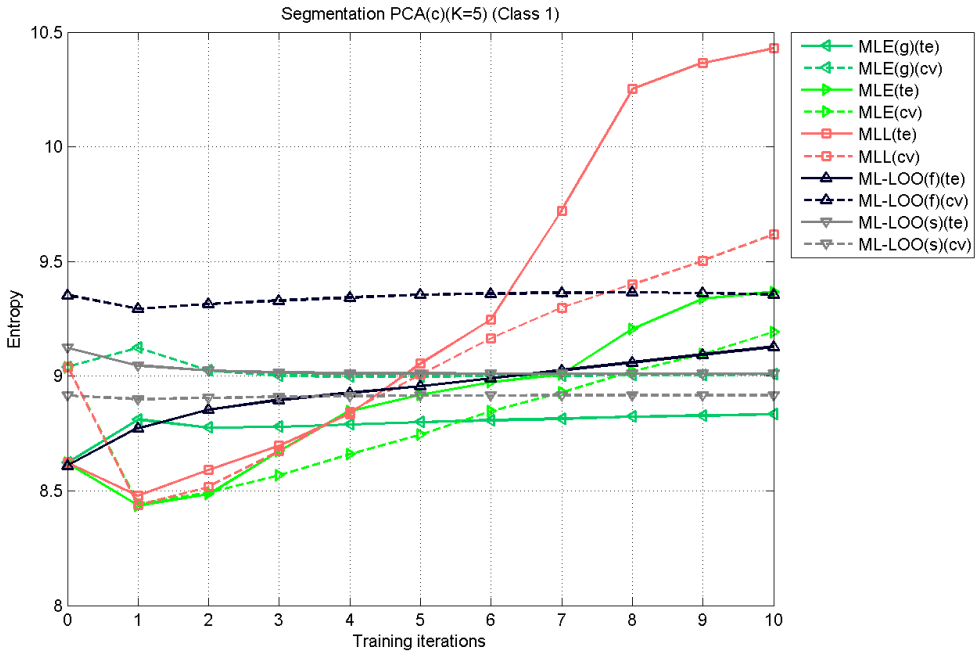

Figure 6.17: *Segmentation entropy results for training iterations (Class 1, CSPEC, PCA5)*

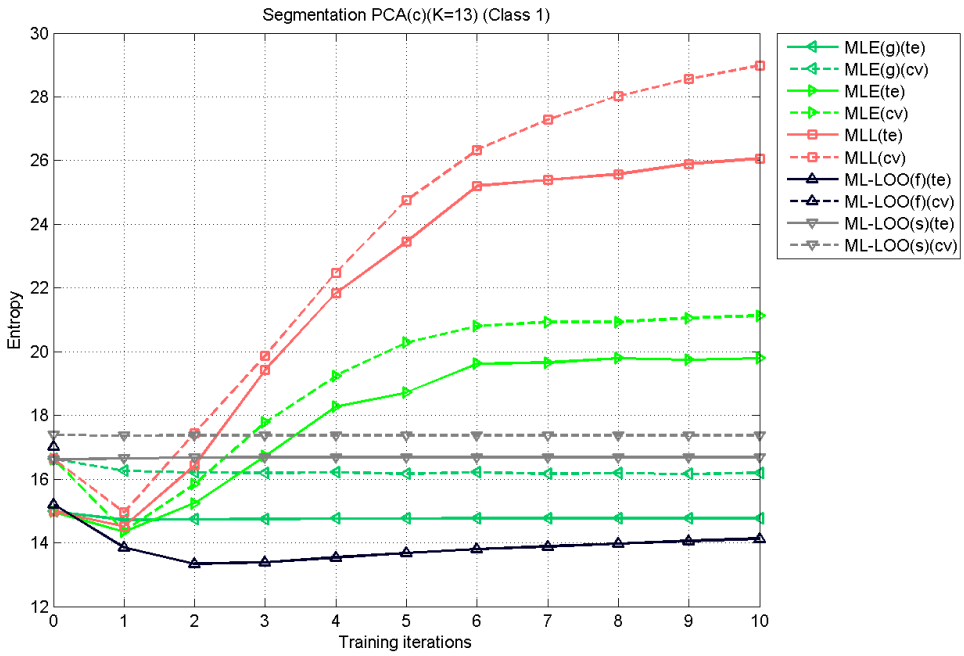

Figure 6.18: *Segmentation entropy results for training iterations (Class 1, CSPEC, PCA1)*

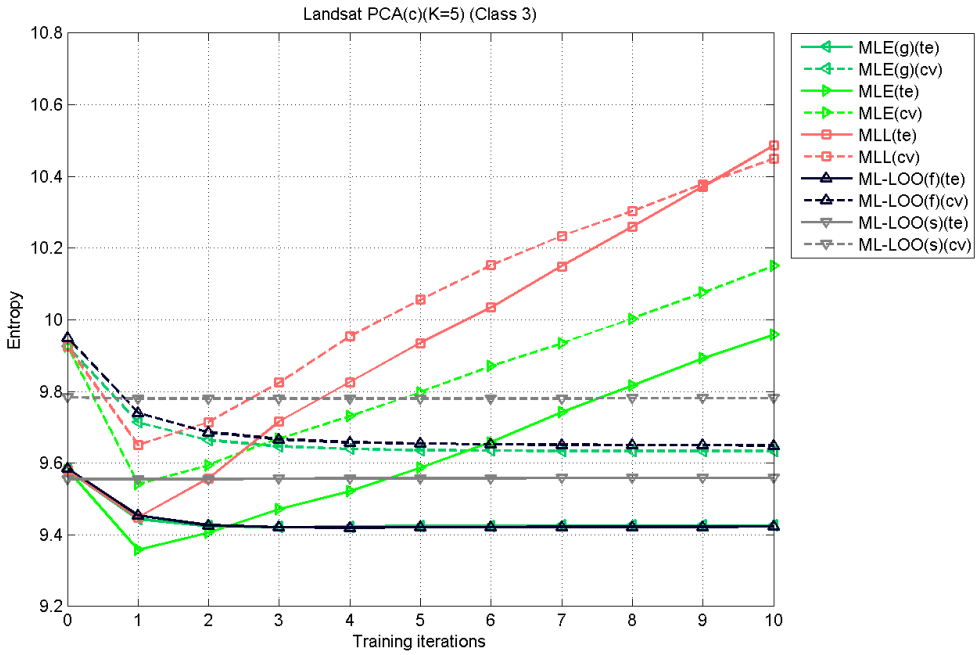

Figure 6.19: *Landsat entropy results for training iterations (Class 3, CSPEC, PCA5)*

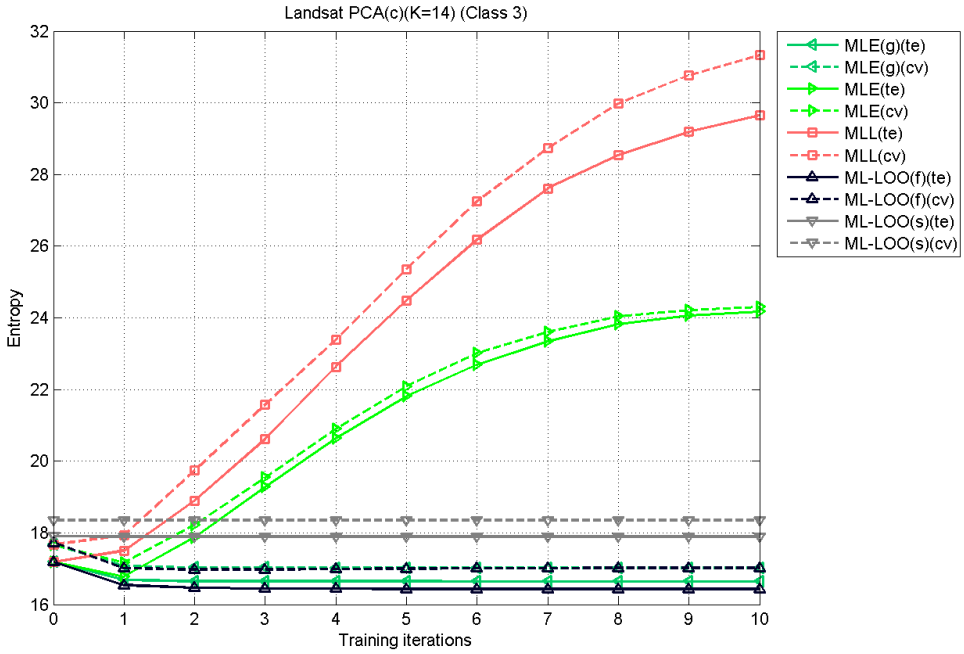

Figure 6.20: *Landsat entropy results for training iterations (Class 3, CSPEC, PCA1)*

one iteration. The entropy of the ML-LOO(s) estimator increases drastically on the first training iteration, and thereafter decreases very slowly as the training continues. The CV entropy of the ML-LOO(f) and MLE(g) estimators converges more slowly to the optimal entropy.

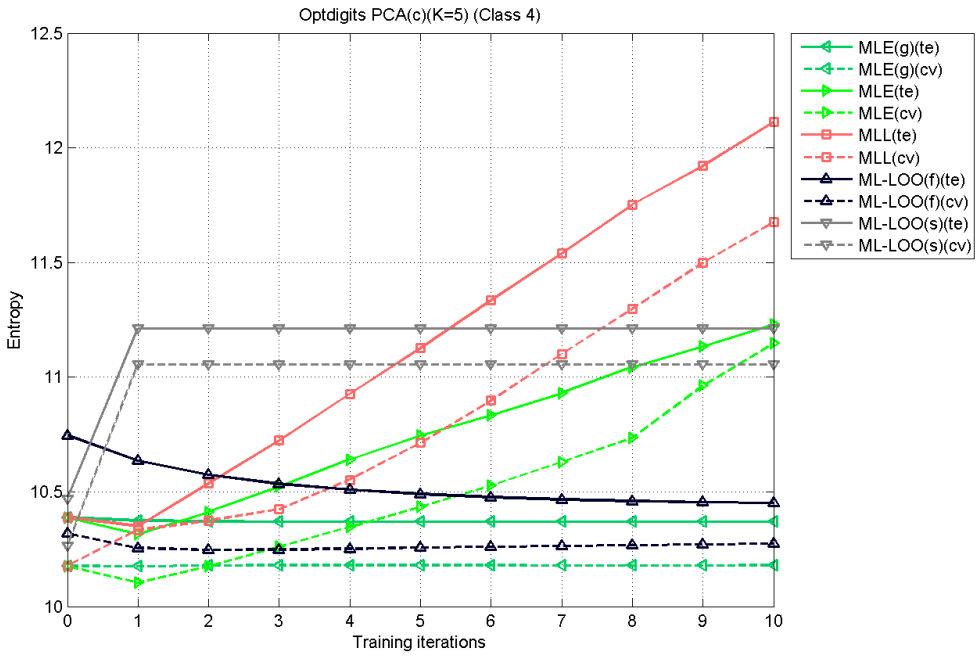

Figure 6.21: *Landsat entropy results for training iterations (Class 4, CSPEC, PCA5)*

Figure 6.22 shows the class-specific entropy results for class 4 of the 50-dimensional PCA1 *Optdigits* dataset. We observe that the optimal MLL and MLE entropy scores are at initialisation, and that the other estimators converge to the optimal entropy more slowly, with the exception of the ML-LOO(f) estimator that experiences an increase in test entropy at iteration 2.

The class-specific entropy results of class 13 of the PCA5 *Isolet* dataset in Figure 6.23 shows that the MLL and MLE estimators converge to the optimal entropy in only one iteration, and that the remaining estimator converges more slowly to the optimal entropy.

Figure 6.24 shows the class-specific entropy results for class 13 of the 123-dimensional PCA1 *Isolet* dataset. We observe that the optimal MLL and ML-LOO(f) entropy is at initialisation and that the optimal MLE and MLE(g) entropy is reached after one training iteration.

In summary, we have observed in Figures 6.15-6.24, that the MLL and MLE estimators converged to the optimal entropy very rapidly (mostly in one iteration), and

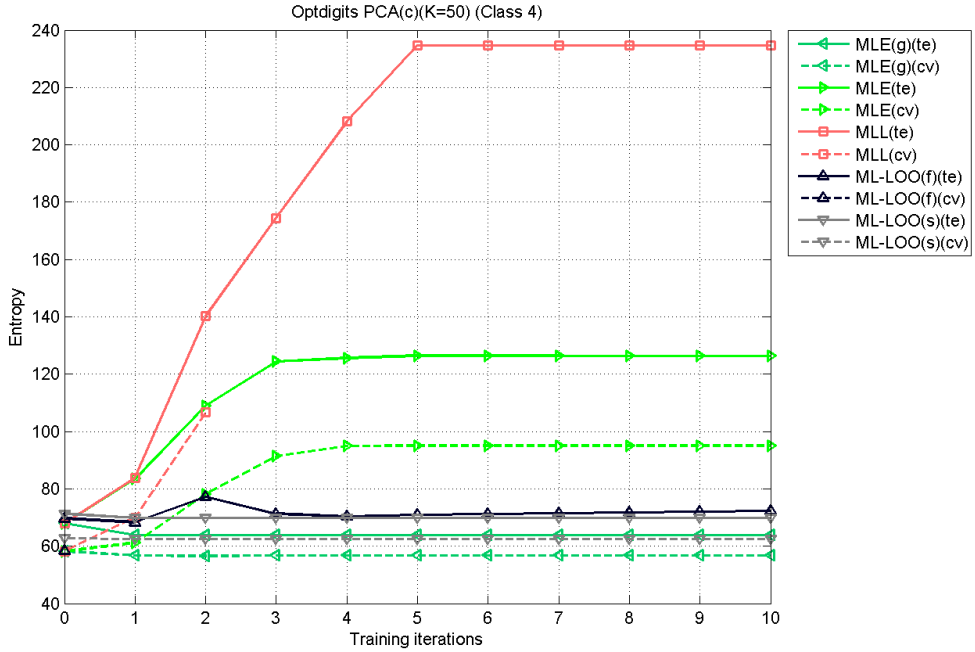

Figure 6.22: *Landsat entropy results for training iterations (Class 4, CSPEC, PCA1)*

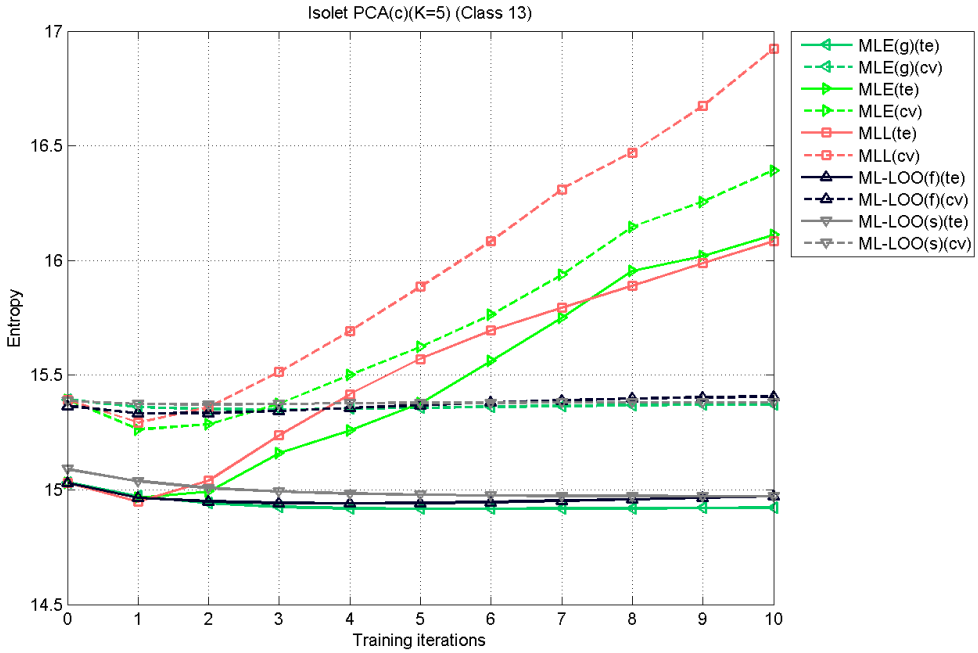

Figure 6.23: *Landsat entropy results for training iterations (Class 13, CSPEC, PCA5)*

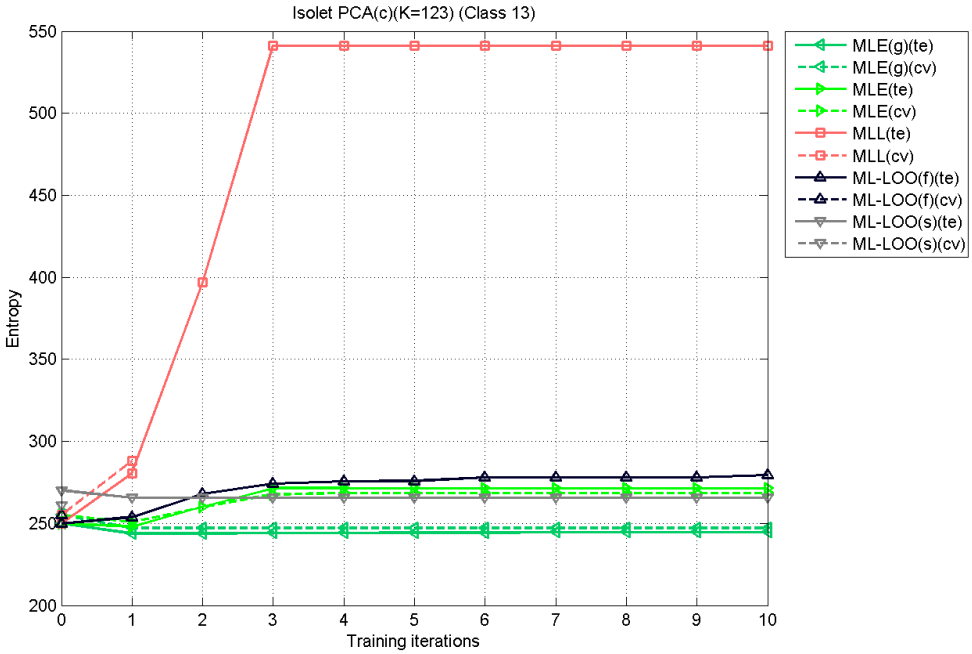

Figure 6.24: *Landsat entropy results for training iterations (Class 13, CSPEC, PCA1)*

that they suffered a severe performance degradation if training continued after the optimal entropy had been reached. We also observed that the other estimators converged more slowly, and did not suffer such severe performance degradations after the optimal number of training iterations had been reached.

6.5 ANALYSIS OF MLE AND MLE(G) BANDWIDTHS

The performance results that we have measured point to a complicated set of relationships between the different estimators. To gain a better understanding of these relationships; we investigate the comparative bandwidths estimated by the MLE and MLE(g) estimators. To this end, we calculate the average and standard deviation of the MLE bandwidths per dimension for each optimal training model of the RW datasets. We then compare the average MLE bandwidths and standard deviations to the MLE(g) bandwidths across dimensions by visually illustrating these calculations for both the optimal PCA5 and PCA1 training models of the RW datasets in Figures 6.25-6.34. We represent the MLE bandwidth information in blue and indicate the average bandwidth per dimension with "x"; we also indicate the standard deviation intervals for each average bandwidth value. The MLE(g) information is indicated in red and we denote the MLE(g) bandwidth per dimension with "o". We denote dimensions with "D" and

bandwidth values with "BW".

We observe in Figures 6.25-6.34 that the average MLE bandwidths and MLE(g) bandwidths generally reduce as dimensionality increases. The dimensions on the x-axes are ordered by principal components such that the d^{th} dimension represents the d^{th} principal component; consequently the eigenvalue of the d^{th} principal component will be equal to the variance of the d^{th} dimension. The variance of the dimensions thus decreases as dimensionality on the x-axis increases (since the eigenvalues decrease as dimensionality increases). The decrease in variance thus leads to a general trend of reduction in average bandwidth as dimensionality increases.

Figure 6.25 shows that for the *Letter* PCA5 data the average MLE bandwidths per dimension are in most cases higher than the MLE(g) bandwidths. Figure 6.26, however, shows that as dimensionality increases, the average MLE bandwidths per dimension decrease relative to the MLE(g) bandwidths per dimension, such that the MLE(g) bandwidths are greater than the average MLE bandwidths for dimensions of 10 and higher. We also observe in Figures 6.25-6.26 that all the MLE(g) bandwidths fall within one standard deviation of the MLE bandwidth means.

We have illustrated with some examples in Section 6.2.3 that as the dimensionality of the datasets increases from PCA5 to PCA1, the number of parameters to estimate increases significantly for the MLE estimator. The reduction in average bandwidth per dimension for the MLE estimator relative the MLE(g) estimator as dimensionality increases might thus be explained by overfitting due to the increasing number of parameters to estimate.

Figure 6.27 shows that for the *Segmentation* PCA5 dataset there is no clear decrease in the average MLE bandwidth per dimension relative to the MLE(g) bandwidth per dimension. Similarly, there is no clear trend for the PCA1 dataset in Figure 6.28. We also observe in Table 6.7 and Table 6.8 that the MLE estimator does not undergo a severe performance degradation when dimensionality is increased from PCA5 to PCA1, which suggests that MLE estimator does not suffer from overfitting for the Segmentation dataset, and therefore we also do not observe a relative decrease in the average MLE bandwidths relative to the MLE(g) bandwidths as dimensionality increases. As with the Letter dataset results, we again observe in Figures 6.27-6.28 that all the MLE(g) bandwidths fall within one standard deviation of the MLE bandwidth means.

Figures 6.29-6.30 show that for the *Landsat* PCA5 and PCA1 datasets, the average MLE bandwidth for the principal component is always greater than the MLE(g) bandwidth; conversely, the MLE(g) bandwidth is always greater than the average MLE

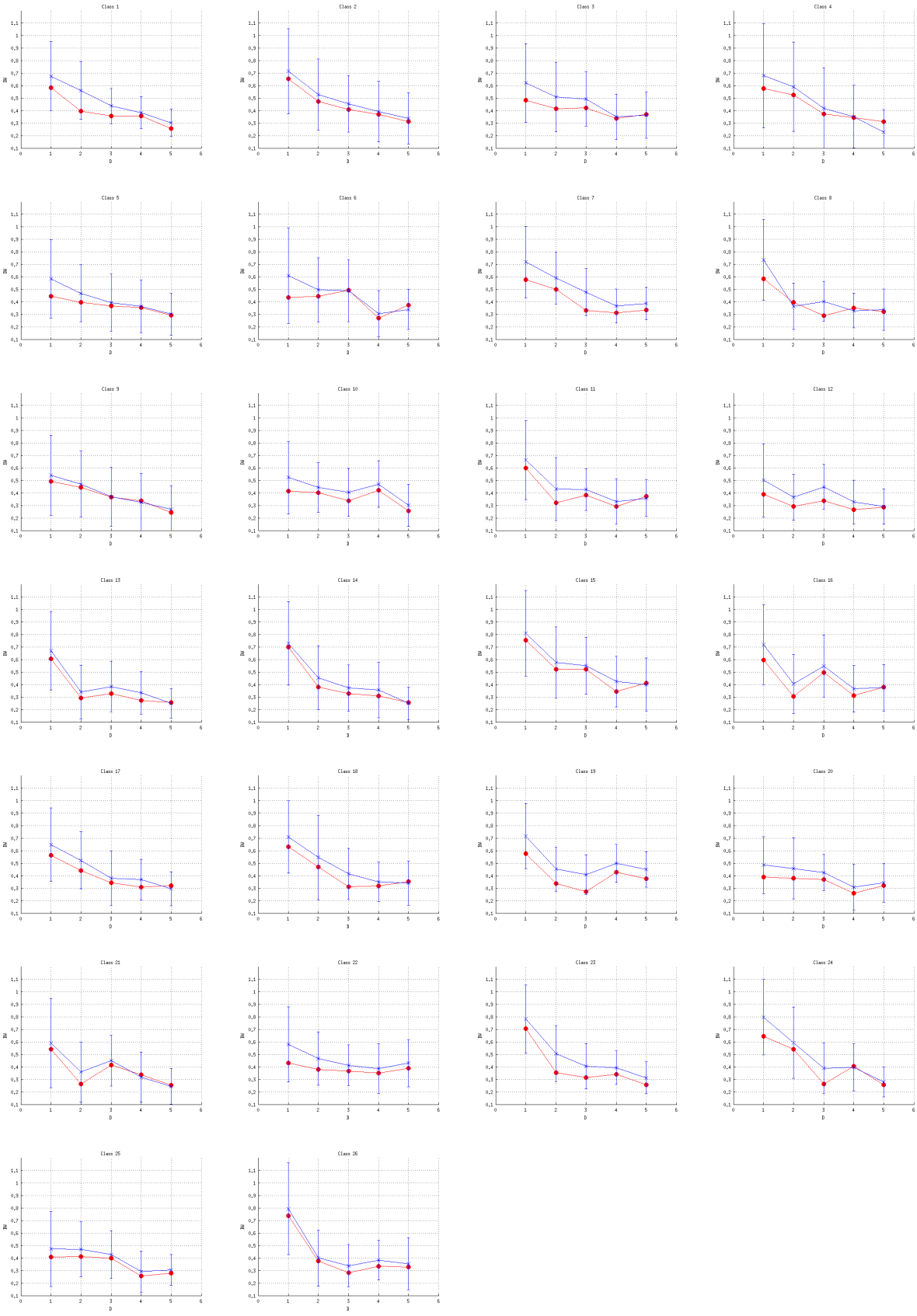

Figure 6.25: *Letter MLE and MLE(g) bandwidths (class-specific PCA5)*

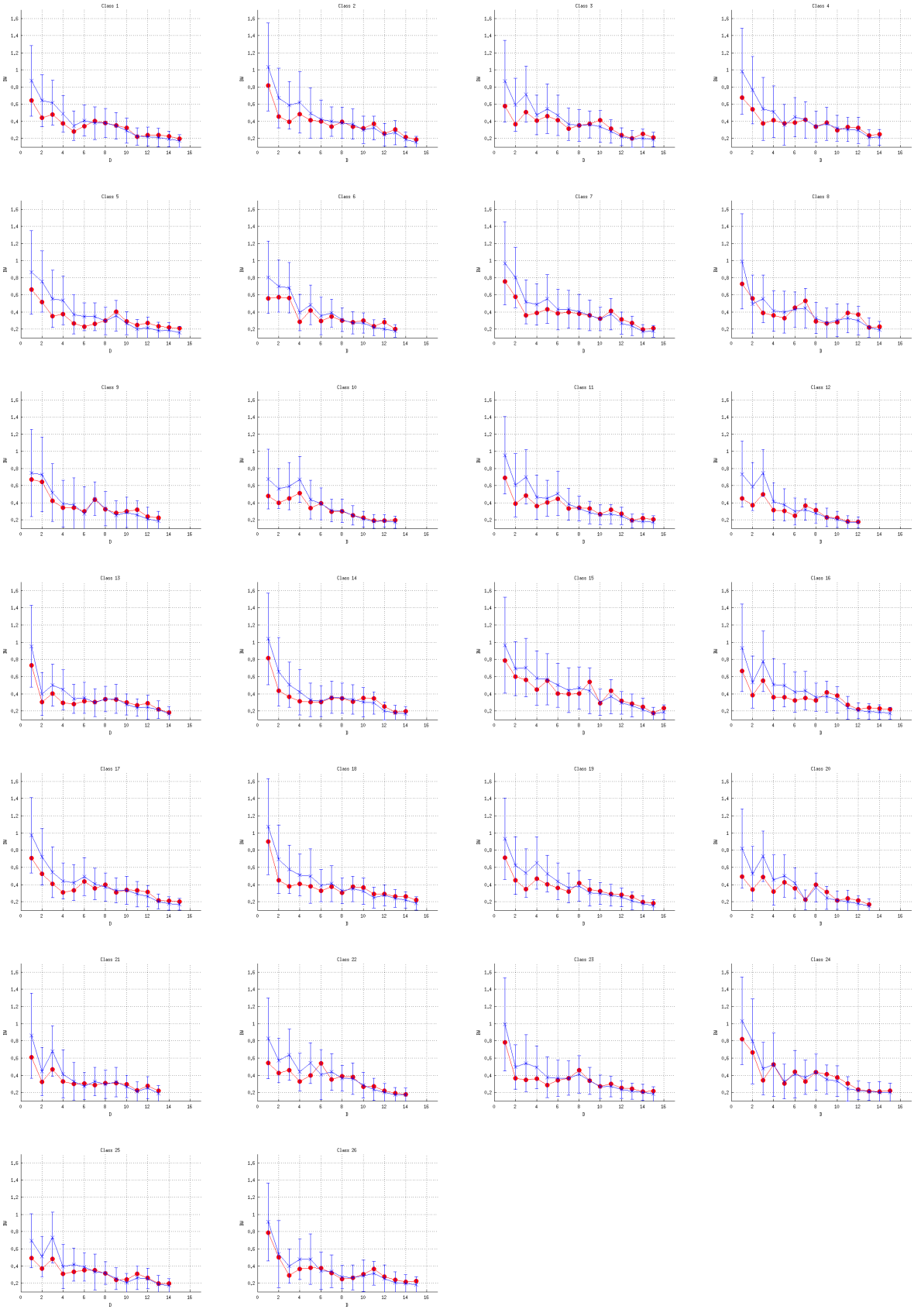

Figure 6.26: *Letter MLE and MLE(g) bandwidths (class-specific PCA1)*

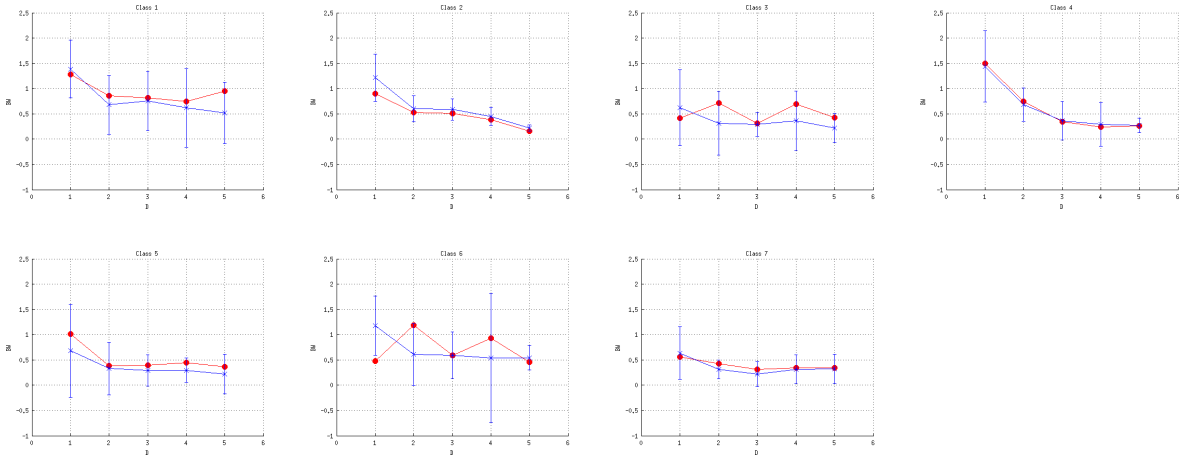

Figure 6.27: *Segmentation MLE and MLE(g) bandwidths (class-specific PCA5)*

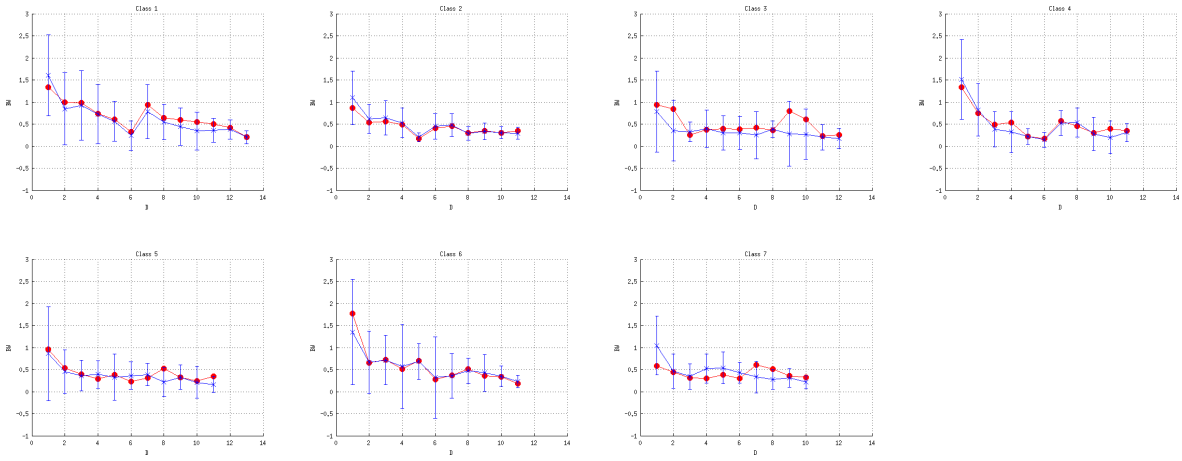

Figure 6.28: *Segmentation MLE and MLE(g) bandwidths (class-specific PCA1)*

bandwidth for the fifth dimension in the PCA5 dataset and the highest dimension for the PCA1 dataset. This observation again implies that generally the average MLE bandwidth is larger than the MLE(g) bandwidth for lower dimensions, and that the MLE bandwidth becomes smaller relative to the MLE(g) bandwidth as dimensionality increases; this suggests that overfitting of the MLE bandwidths increases as dimensionality increases. The results in Table 6.11 and 6.12 show that the MLE estimator performs optimally on four of the six classes for the PCA5 dataset and experiences a relative degradation in performance since it performs optimally on only two of the six classes on the PCA1 dataset. These results seem to support to possibility of overfitting on the PCA1 data due to the higher number of parameters to estimate.

We also observe in Figures 6.29-6.30 that most of the MLE(g) bandwidths fall within one standard deviation of the MLE bandwidth means, with four exceptions on the Landsat PCA1 data and two exceptions of the Landsat PCA5 data. These large

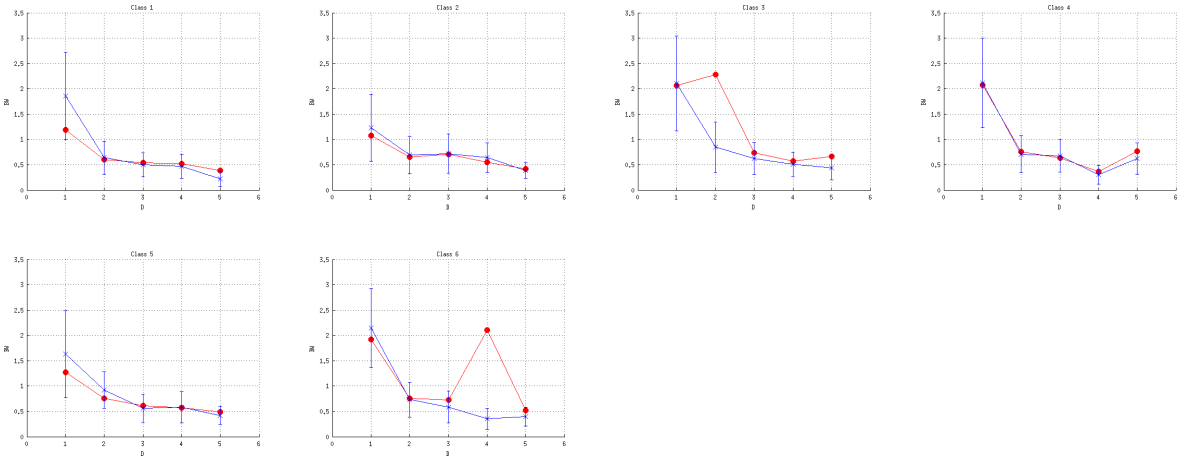

Figure 6.29: *Landsat MLE and MLE(g) bandwidths (class-specific PCA5)*

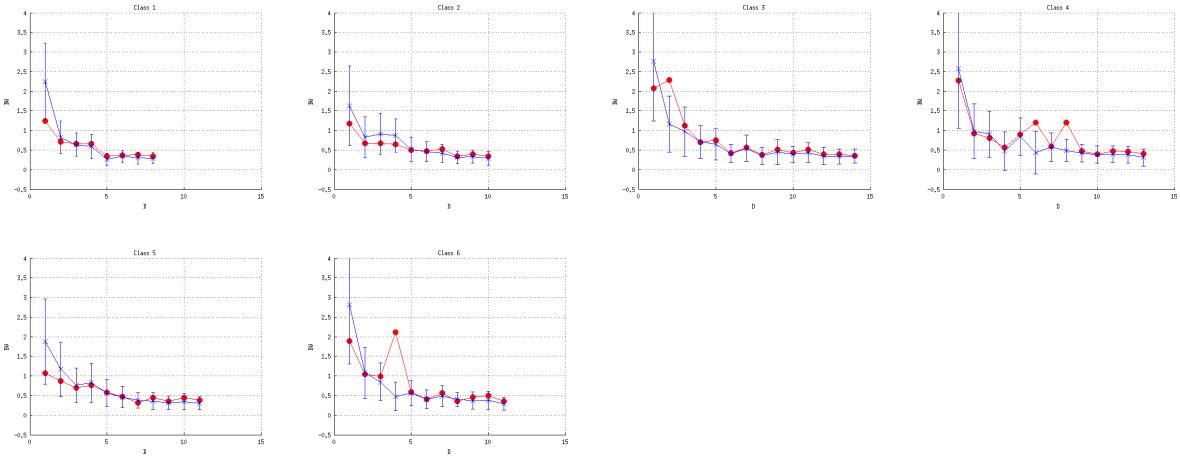

Figure 6.30: *Landsat MLE and MLE(g) bandwidths (class-specific PCA1)*

bandwidths for the MLE(g) bandwidth estimates are caused by some data points in the training set that are very far from the remaining data points in a certain dimension. A single data points that is far removed from the remaining data points in a certain dimension can cause the bandwidth in that dimension to become extremely big in order to maximise the ML LOUT criteria. This is one inherent drawback of the ML LOUT criteria if an identical bandwidth is estimated for all kernels; fortunately this drawback does not apply to the MLE estimator since the bandwidths for kernels are adjusted individually and a training data point that is far removed from all other data points in a certain dimension will have a large bandwidth in that dimension for only the specific kernel; the bandwidths of the remaining kernels far from the outlier data points will not be susceptible to over smoothing.

The *Optdigits* PCA5 bandwidth results in Figure 6.31 show that the average of the MLE bandwidths per dimension and the MLE(g) bandwidths have very similar trends

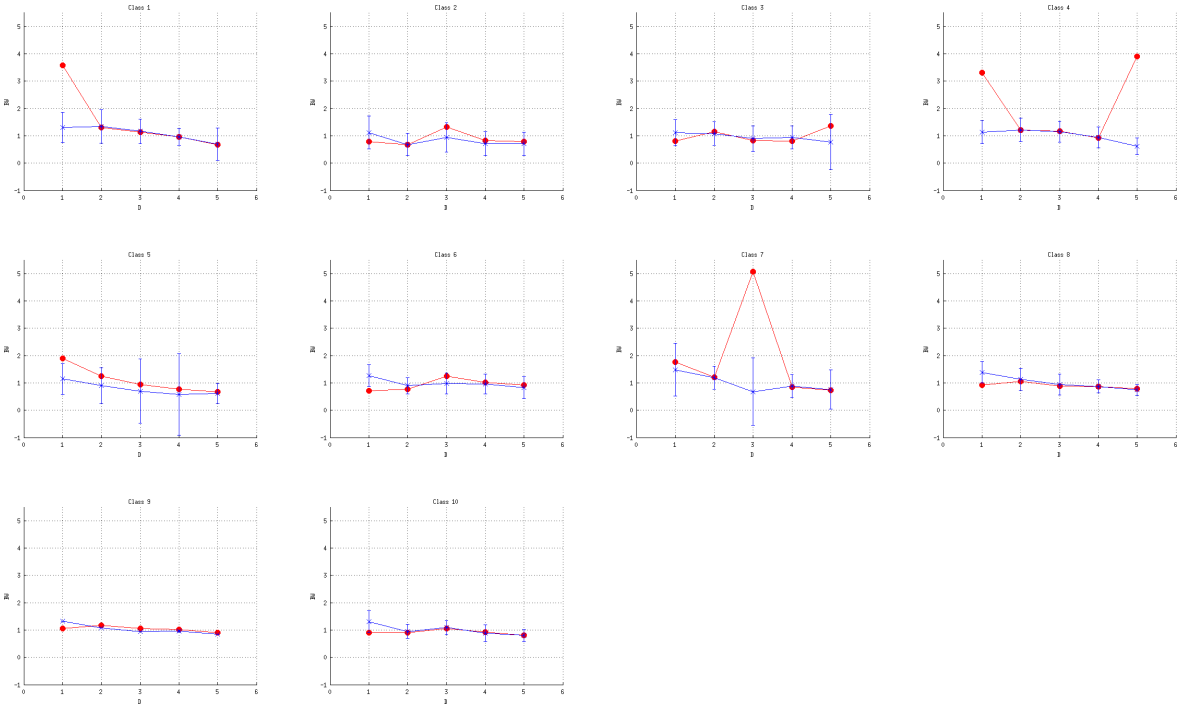

Figure 6.31: *Optdigits MLE and MLE(g) bandwidths (class-specific PCA5)*

with the exception of a few large bandwidth outliers e.g. class 1, dimension 1; class 4, dimensions 1 and 5 and class 7, dimension 3. Similar to the MLE(g) bandwidth outliers in the Landsat dataset, the large bandwidths for the MLE(g) bandwidth estimates are caused by some data points in the training set that are very far from the remaining data points in the specific dimension.

The *Optdigits* PCA1 bandwidth results in Figure 6.32 show that the standard deviations of the MLE bandwidths in all dimensions for all classes are zero. We confirmed that for these classes all kernels are identical in all dimensions, which results in a zero standard deviation of bandwidths in all dimensions. Zero standard deviation of the MLE bandwidths for a class in all dimensions are caused in cases where the initial Silverman kernel bandwidths (which are identical for all kernels in a dimension) yield a better entropy score than the updated bandwidths (see, for example, Figure 6.22). This will cause all the initial kernel bandwidths in a class to be identical since Silverman initialisation is used, which only estimates a unique bandwidth per dimension and makes use of the same bandwidth for each dimension across all kernels (see Section 6.2.6 for a more detailed explanation). If the bandwidths for all kernels are identical at initialisation and are not updated, the standard deviation of the bandwidth for each dimension will be zero.

Zero standard deviations of the MLE bandwidths for only certain dimensions within

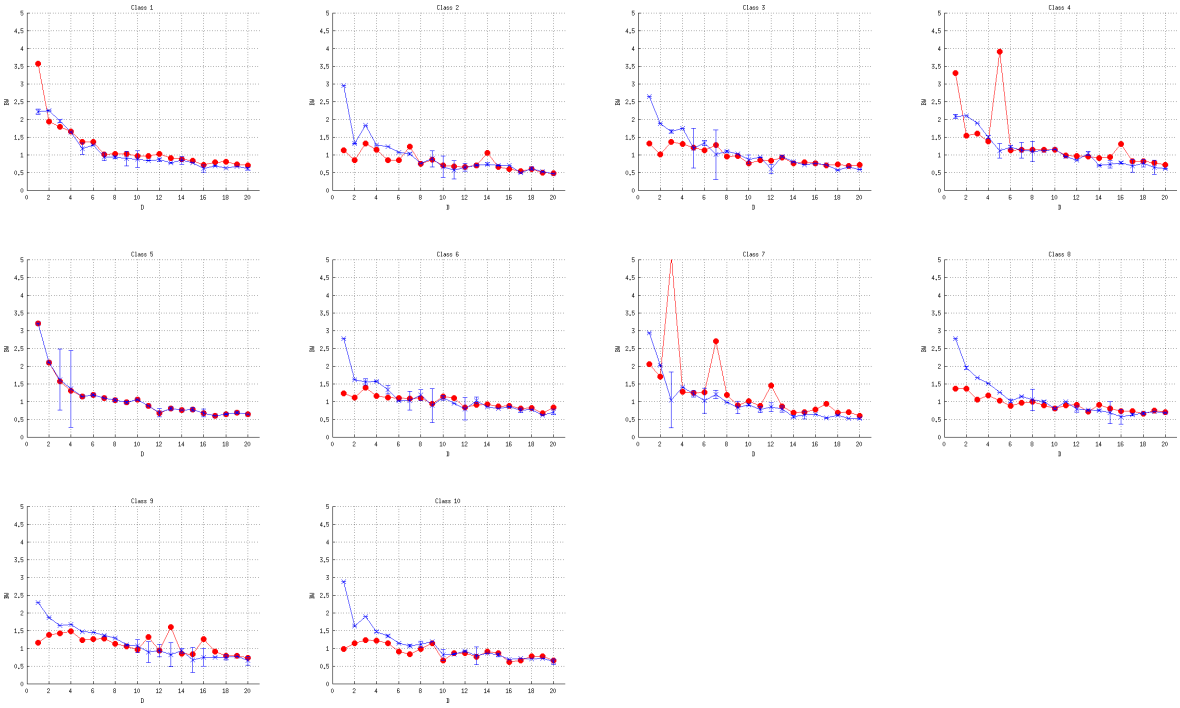

Figure 6.32: *Optdigits MLE and MLE(g) bandwidths (class-specific PCA1)*

a class are encountered in cases where the initial bandwidths do not result in the optimal entropy; the optimal entropy is reached after the bandwidths are updated, and bandwidth regularisation is performed on all of the bandwidths for the dimensions with zeros standard deviation. After each iteration MLE regularisation is performed by comparing the updated bandwidths to their respective minimum bandwidths, which is calculated for each kernel as the distance to its nearest neighbour (with non-zero distance) per dimension. If the BW of a kernel is below this minimum threshold in any of the dimensions, the bandwidths in those dimensions are replaced with the initial Silverman bandwidths. The dimensions with zero standard deviation bandwidths are thus examples where the bandwidths of all kernels in those dimensions were below their respective minimum bandwidths, and were replaced with the initial Silverman bandwidths. The dimensions that do not have bandwidths with zero standard deviations are the dimensions where not all the bandwidths went below their minimum threshold and were thus updated and different from the initial Silverman bandwidths.

We observe, for example, that the standard deviations of the MLE bandwidths for the Optdigits, PCA1 class 1 dataset are zero. For this class the Silverman initial bandwidths were thus the optimal bandwidths. In, for example, class 2 we see that the MLE bandwidths of most dimensions have zero standard deviation, which shows that all the bandwidths in these dimensions were identical and equal to their initial

Silverman bandwidths (since they underwent regularisation). Dimensions 9, 10 and 11 have non-zero standard deviations, which show that some of the bandwidths in these dimensions were updated without going below their respective minimum bandwidths; thus not all the bandwidths in these dimensions underwent regularisation. We have verified that the optimal bandwidths for this class were identical in all dimensions, except dimensions 9, 10 and 11.

We also observe that for the Optdigits, PCA1 class 5 data, the average MLE bandwidths and MLE(g) bandwidths are identical for all dimensions except dimensions 3 and 4. Since the standard deviations of the MLE bandwidths for dimensions 3 and 4 are non-zero, this implies that some of the kernels were updated in these dimensions. The remaining dimensions have zero standard deviation, which shows that they were not updated after initialisation. The MLE(g) bandwidths are identical to the average MLE bandwidths for the dimensions with zero standard deviation since the MLE(g) estimator optimal bandwidths were the initial bandwidths. The MLE and MLE(g) estimators use the same initial Silverman bandwidths, and therefore the bandwidths are identical for all dimension that were not updated. Table 6.16 shows that for this class the MLE and MLE(g) estimators have identical entropy values, since they have the same optimal bandwidths.

In summary, for classes where the standard deviations of the MLE bandwidths for all dimensions are zeros, the optimal BWs for those classes were at initialisation. For classes that have MLE bandwidths with zero standard deviation in only some dimensions, all kernels have undergone regularisation in the dimensions with zero standard deviations. The dimensions with non-zero standard deviations have not undergone regularisation on all kernels in that dimension, but may have undergone regularisation on some kernels in that dimension.

The *Isolet*, PCA5 dataset bandwidth results in Figure 6.33 show that the optimal bandwidths for class 1 are the initial Silverman bandwidths since the standard deviations of the MLE bandwidths for all dimensions are zero. classes 3, 5, 9, 10 and 14 have zero standard deviation in some dimensions; for these classes all MLE bandwidths underwent regularisation in the dimensions for which the MLE bandwidth standard deviations are zero; the bandwidths in these dimension where thus reset to the initial Silverman bandwidths.

The *Isolet*, PCA1 bandwidth results in Figure 6.34 show that for class 11 the initial bandwidths were the optimal bandwidths for the MLE estimator, since the MLE bandwidths have zero standard deviations in all dimensions. For classes 3, 10, 14, 17, 20 and 26 all the MLE bandwidths underwent regularisation in the dimensions with

zero standard deviations which reset the bandwidths in these dimension to their initial Silverman bandwidths. The dimensions with non-zero standard deviations, such as class 3, dimension 11 might have undergone regularisation on some of the bandwidths but not all of the bandwidths. The bandwidths that did not undergo regularisation in dimension 11 were thus updated and are different to the initial bandwidths which results in a non-zero standard deviation for this dimension.

The analysis of the MLE and MLE(g) bandwidths in this section have shown that valuable insight can be obtained by visualising the average and standard deviations of the MLE bandwidths and comparing these bandwidths to the MLE(g) bandwidths across dimensions.

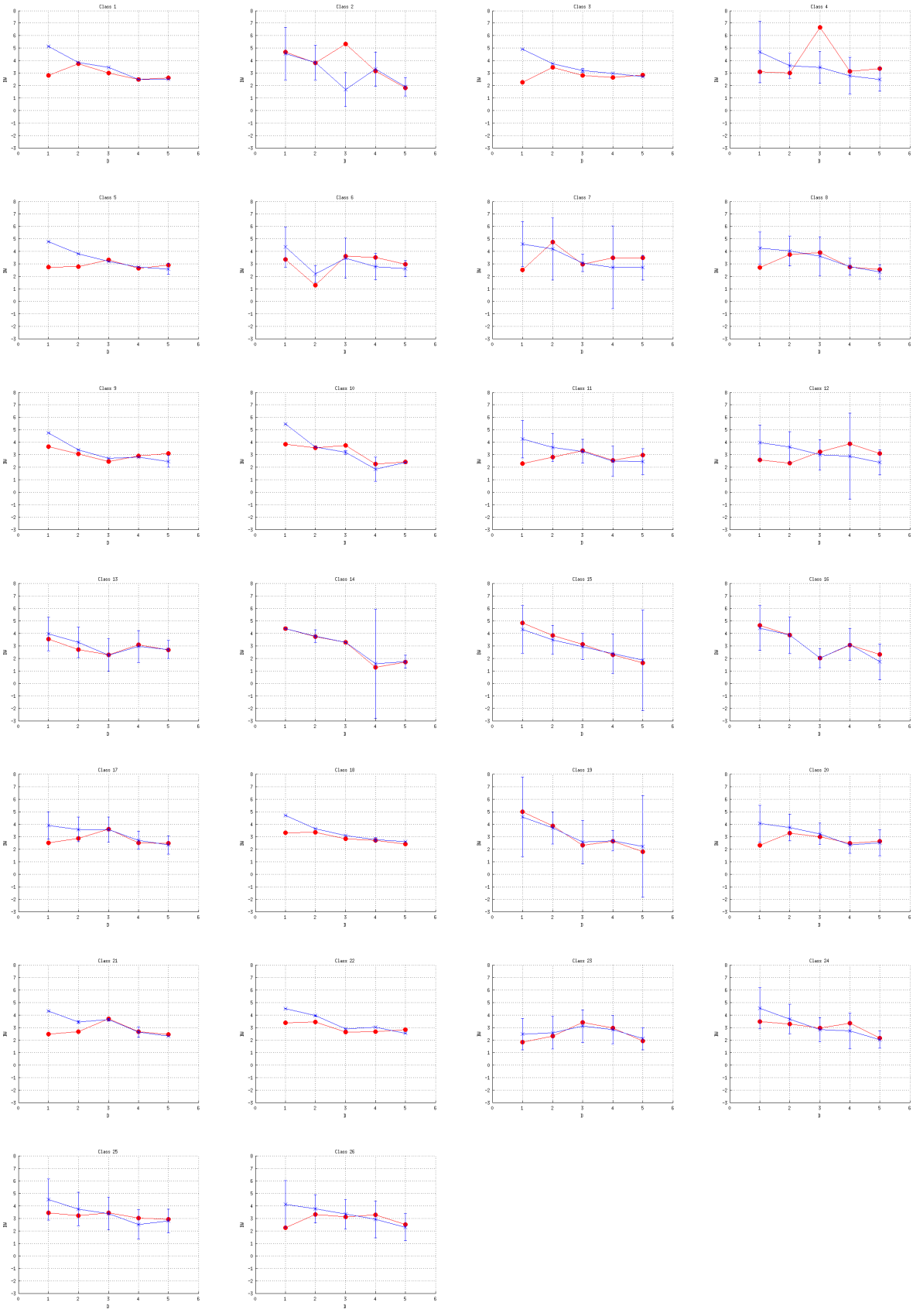

Figure 6.33: *Isolet MLE and MLE(g) bandwidths (class-specific PCA5)*

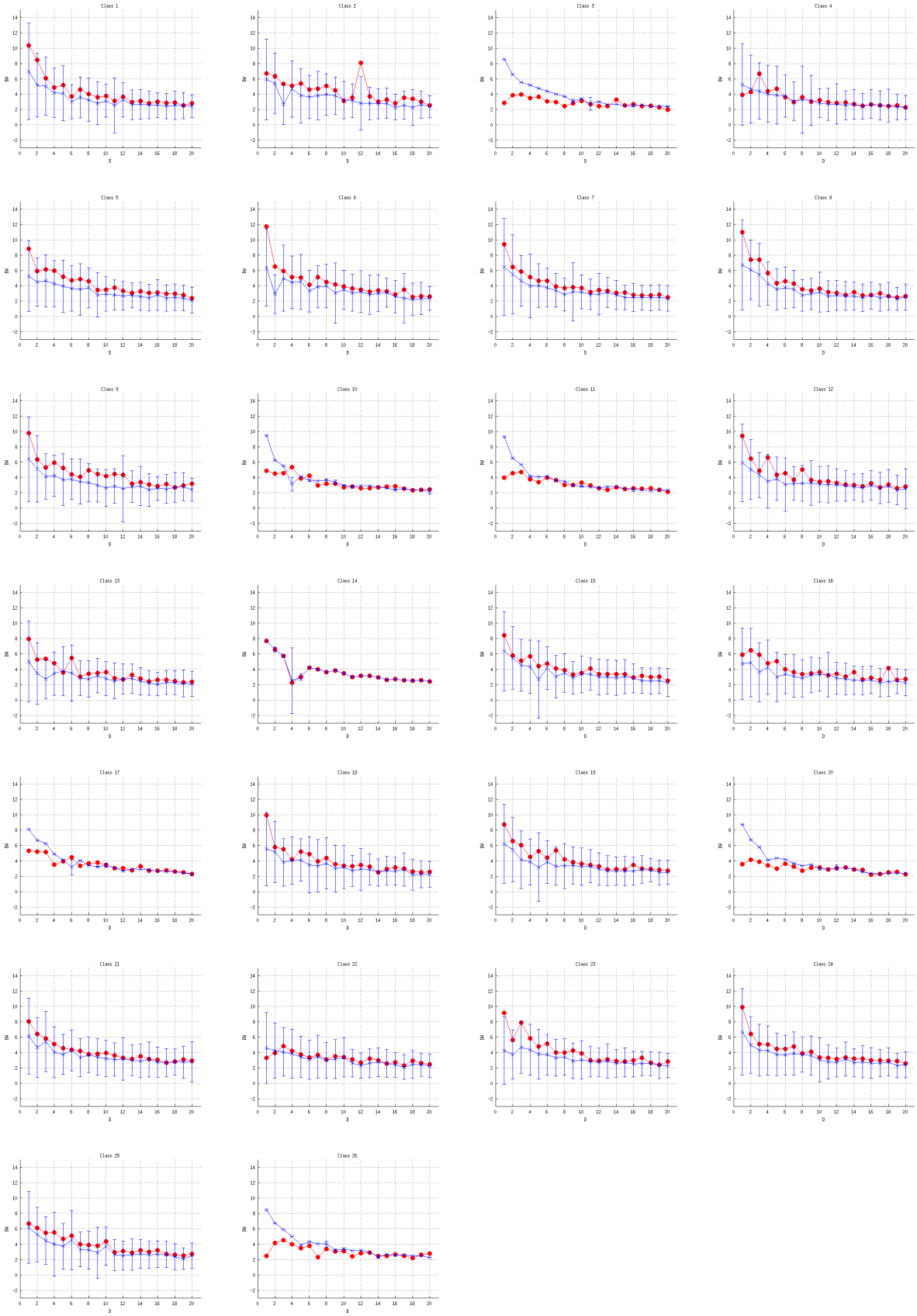

Figure 6.34: *Isolet MLE and MLE(g) bandwidths (class-specific PCA1)*

6.6 CONCLUSION

In this chapter we compared the performance of iterative ML bandwidth estimators on artificial, CV and RW datasets. The artificial data experiments were very informative, since the properties of the artificial data were known and allowed us to investigate the behaviour of the estimators under different conditions.

On the unimodal artificial data we observed that the MLL and MLE estimators underperformed for dimensionalities higher than 3; these estimators seem to suffer more from the curse of dimensionality since they estimate more parameters. These estimators estimate a unique bandwidth matrix for each kernel, while the other ML estimators estimate an identical bandwidth matrix per kernel. The MLE(g), ML-LOO(s) and ML-LOO(f) estimators performed well on the standard MVN data. When scale variations between features were introduced, the ML-LOO(s) estimator underperformed since the same bandwidth is used for all dimensions. When correlation was introduced between features the ML-LOO(f) estimator performed optimally, and the relative performance of this estimator improved further as dimensionality was increased.

On the bimodal artificial data we observed that the ML-LOO(f) and ML-LOO(s) estimators underperformed for high dimensionalities, and that the relative performance of the MLL and MLE estimators improved as dimensionality increased, for bimodal data that consist of two MVN distributions. When we introduced scale variations within features, the MLL and MLE estimators performed optimally for all dimensions, and their relative performance improved as dimensionality increased. When correlation was introduced into the data (with identical covariance matrices for the two modes), the ML-LOO(f) estimator performed optimally on all dimensions. When we introduced anisotropy by using different covariance matrices for the two modes, the ML-LOO(f) estimator performed optimally in most dimensions; the other estimators were, however, more competitive.

On the PCA5 CV data, the MLE(g) estimator performed well on all classes of the Iris and Waveform datasets. For the PCA5 CV data the MLE estimator performed well on some classes of the Old Faithful, Balance Scale, Iris, Diabetes and Vehicle datasets. The MLL estimator performed well on some classes of the Old Faithful, Balance Scale, Diabetes, and Vehicle datasets. The ML-LOO(f) estimator performed well on some classes of the Balance Scale, Diabetes, Vehicle and Ionosphere datasets, and optimally on both classes of the Heart-statlog dataset. The ML-LOO(s) estimator performed well on the Waveform dataset, and on one class of the Ionosphere dataset. The ML-LOO(s) estimator underperformed on the Old Faithful, Balance Scale and Iris datasets, and on

some classes of the Diabetes, Heart-statlog and Vehicle datasets.

On the PCA1 CV data, the MLE(g) estimator performed optimally on the Waveform dataset and well on some classes of the Balance Scale, Iris, Vehicle and Ionosphere datasets. The MLE estimator performed optimally on the Ionosphere dataset, and on some classes of the Balance Scale, Iris and Vehicle datasets. The MLL estimator performed well on some classes of the Balance Scale dataset. The ML-LOO(f) estimator performed optimally on the Diabetes and Heart-statlog datasets and well on the Waveform and some classes of the Vehicle, Iris and Balance Scale datasets. The ML-LOO(s) did not perform well on any of the classes for all datasets, it underperformed on the Old Faithful, Balance Scale and Iris datasets, and underperformed on some classes of the Diabetes, Vehicle, Waveform and Ionosphere datasets.

On the *Letter* dataset the ML-LOO(f) estimator performed optimally on most classes for both PCA5 and PCA1 data, while the ML-LOO(s) underperformed on most classes of the PCA5 and PCA1 data. On the *Segmentation* dataset, the MLE estimator performed optimally on most classes, followed by the MLL and the ML-LOO(f) estimator underperformed on most classes of the PCA5 data, while the ML-LOO(s) underperformed on most classes of the PCA1 data. On the *Landsat* dataset, the MLE estimator performed optimally on most classes of the PCA5 data, and the ML-LOO(f) estimator performed optimally on most classes of the PCA1 data. The ML-LOO(s) estimator underperformed on all classes of both the PCA5 and PCA1 data. On the *Optdigits* dataset, the ML-LOO(f) estimator performed optimally on most classes of the PCA5 data, and the MLE(g) estimator performed optimally on most classes of the PCA1 data. The MLE and ML-LOO(s) estimators underperformed on most classes of the PCA5 data, and the ML-LOO(f) and ML-LOO(s) estimators underperformed on most classes of the PCA1 data. On the *Isolet* dataset the MLE and MLE(g) estimators performed optimally on most classes on the PCA5 data, and the MLE(g) performed optimally on most classes of the PCA1 data. The ML-LOO(s) underperformed on most classes of the PCA5 data, and underperformed on all classes of the PCA1 data.

In summary, the results in this chapter indicate that none of the kernel bandwidth estimators performed optimally on all tasks, and that the optimal estimator should be selected based on the following data properties: scale variation between features, scale variation within features, and correlation between features.

The artificial dataset experiments indicated that the MLE(s) estimator should be selected if there are no significant scale variations between and within features, and if there is no correlation between features. The ML-LOO(f) estimator should be selected

if there are no scale variations within features, and if there is correlation between features. The MLE(g) estimator should be selected if there are no scale variations within features, and if the features are uncorrelated. The MLL and MLE estimators should be selected if there are scale variations within features and no correlation between features, or if there are scale variations within features and the correlation between features is removed with a class-specific PCA transformation.

Since these data properties are not easily measured and quantified from data, it is not always clear which estimator will perform optimally on a given task. We have shown that we can empirically compare the performance of these estimators by making use of either CV or test set entropy to determine the optimal estimator for a given task.

We have shown that the newly introduced MLE(g) and MLE estimators are valuable, since they provide researchers and practitioners with two alternative estimators to employ, which might perform optimally under the conditions mentioned above. We have also shown that the MLE and MLL estimators converge rapidly, and in many cases converge to the optimal entropy in only a single iteration. This fast convergence performance of these estimators presents two advantages over the other ML estimators, (1) reduced training times due to fewer training iterations and (2) the fact that they may be used without an iterative bandwidth update procedure; thus if they are initialised with the Silverman rule-of-thumb, an analytical equation can be used to calculate a unique bandwidth matrix for each kernel. The MLE(g) and MLE estimators also have sound theoretical foundations, which provide us with an intuitive understanding of their behaviour.

Chapter Seven

Conclusion

Contents

7.1 Introduction and Thesis Summary 158

7.2 Summary of Findings 160

7.3 Summary of Contributions 162

7.4 Suggestions for Further Research 163

7.5 Conclusion 165

7.1 INTRODUCTION AND THESIS SUMMARY

In Chapter 1 we stated that the behaviour of conventional density estimators on high-dimensional pattern recognition tasks is not fully understood, and has only recently been investigated in the context of such high-dimensional spaces for the purpose of pattern recognition [17]. We also stated that recent advances in ML kernel density estimation have shown that KDEs hold much promise for estimating nonparametric density functions in high-dimensional spaces, and that two notable kernel bandwidth estimators were recently introduced to the field of pattern recognition, namely the MLL and ML-LOO estimators. These estimators have proven to be effective at the task of kernel bandwidth estimation on real-world pattern-recognition tasks. Both estimators were independently derived from the ML theoretical framework, the only

differences being the simplifying assumptions made in their derivations to limit the complexity of the bandwidth optimisation problem. For both estimators, it was shown that non-parametric kernel density estimation can be performed in high-dimensional spaces with a reasonable degree of success [17, 18]. However, the practical implications of the simplifying assumptions made in the derivations of the MLL and ML-LOO estimators on RW density estimation tasks were not understood since a derivation of a kernel bandwidth estimator without these assumptions did not exist.

Therefore the main theoretical objective of this study was to derive an ML kernel bandwidth estimator without the simplifying assumptions imposed by state-of-the-art ML kernel bandwidth estimators; and the main empirical objectives were (1) to gain a better understanding of the behaviour of conventional kernel bandwidth estimators (specifically in the context of high-dimensional feature spaces), and (2) to investigate, on a number of representative pattern recognition tasks, whether the increased flexibility of a more general ML kernel bandwidth estimator will yield an improvement in performance over state-of-the-art ML kernel bandwidth estimators.

In Chapter 2, we presented a brief survey of existing approaches to non-parametric density estimation, and motivated why kernel density estimation is a suitable candidate for this task. Thereafter, in Chapter 3, we explained why non-parametric density estimation in high-dimensional spaces is important in practice, and presented some non-intuitive examples of high-dimensional data properties.

Chapter 4 was devoted to the derivation of two new kernel bandwidth estimation techniques from the ML framework, namely the MLE and global MLE bandwidth estimators and also derived the MLL and ML-LOO estimators from the same theoretical framework. The MLE estimator was derived without the constraints imposed on the MLL and ML-LOO estimators, and therefore fulfilled our theoretical objective. We also proposed a new kernel bandwidth initialisation algorithm, named the MLE rule-of-thumb, since the MLE, MLL and ML-LOO estimators require bandwidth initialisation to initialise the optimisation procedure of the ML objective function with respect to the kernel bandwidths

In Chapter 5, we investigated the performance of conventional kernel bandwidth estimators and the MLE rule-of-thumb kernel bandwidth estimator on a number of representative pattern-recognition tasks, to gain a better understanding of the performance of these estimators, specifically on high-dimensional tasks, and to suggest a suitable bandwidth initialisation algorithm for the ML estimators derived in Chapter 4.

In Chapter 6, we performed simulation studies to compare the performances of

the MLE, global MLE, MLL, spherical covariance ML-LOO and full covariance ML-LOO estimators on artificial, CV and RW data, to gain a better understanding of the conditions under which the different estimators perform optimally, and to investigate whether the increased flexibility of the MLE estimator leads to improved density estimation in practice specifically in high-dimensional spaces.

In this chapter we summarise our findings of the above-mentioned experiments in Section 7.2; we summarise the contribution of this work in Section 7.3; we suggest future work in Section 7.4 and make concluding remarks in Section 7.5.

7.2 SUMMARY OF FINDINGS

In our theoretical derivations we found that the optimal bandwidth expression of the MLE and MLL estimators are very similar, except that the numerator and denominator of the MLE bandwidth expression is normalised with a LOUT likelihood score that serves as a bandwidth regularisation term. We also showed that the new global MLE estimator is a compromise between the spherical ML-LOO and the full covariance ML-LOO estimators. Since an identical diagonal bandwidth matrix is estimated for all kernels it is similar to the spherical ML-LOO estimator, except that the different features are allowed different bandwidths. Therefore the global MLE is similar to the full covariance ML-LOO estimator, except that it does not have non-zero off-diagonal components in the bandwidth matrix. In the comparative study of the conventional kernel bandwidth estimators and the MLE rule-of-thumb kernel bandwidth estimator, we found that the Silverman rule-of-thumb estimator performed consistently across numerous tasks and dimensionalities and therefore recommended that this estimator is a suitable bandwidth initialisation procedure for ML estimators, even in high dimensions. We also noted that the Silverman estimator has an intuitive theoretical motivation, since it minimizes the AMISE under the assumption of Gaussian data distributions, which made the choice of this estimator more appealing. We also found that the relative performance of the MLE rule-of-thumb estimator improved as dimensionality increases, and it was therefore suggested that this estimator should be considered as a suitable estimator for very high-dimensionalities.

In the comparative study of the ML bandwidth estimators we simulated various data properties with artificial data and showed that no estimator is optimal, and that the performance of an estimator is determined by the properties of the dataset. We confirmed this conclusion with our experiments on CV and RW datasets - since no estimator performed optimally on all classes of the datasets considered. We did, however,

find that the full covariance ML-LOO and global MLE estimators often outperformed the MLE and MLL estimators that have more parameters, and that the spherical ML-LOO estimator often underperformed relative to the other estimators by a significant margin.

We have obtained the following insights with respect to the selection of the optimal ML kernel bandwidth estimator based on the intuition gained from our theoretical derivations of the ML estimators and our experiments on artificial data. The spherical ML-LOO estimator cannot accurately model significant scale variations between and within features and no outliers. Since an identical bandwidth matrix is estimated for all kernels, the spherical ML-LOO estimator cannot adapt the bandwidth of kernels throughout feature space to model scale within features and thus cannot adapt bandwidths to be sufficiently wide for outliers. Since an identical bandwidth is used in all dimensions, the spherical ML-LOO estimator can also not adapt kernel bandwidths between dimensions to model scale variations between features. The local density function in the proximity of kernels is also restricted to be parallel to the feature axes since a spherical covariance matrix is used. We note, however, that this does not imply that the spherical estimator cannot model correlation between features to some degree. Since kernels are placed over data points, the density estimate can capture directions of variation that are not necessarily parallel to the feature axes (e.g. if data points are not aligned in direction parallel to the feature space).

The full covariance ML-LOO estimator cannot model significant scale variations within features. Similar to the spherical ML-LOO estimator, the full ML-LOO estimator estimates a single bandwidth matrix for all kernels, and can thus not adapt bandwidths based on local variations in scale. This also implies that his estimator will not be able to model outliers well, since the bandwidth cannot be spread out for kernels centred on data points in low density regions. The full ML-LOO estimator can, however, model local variations in the proximity of kernels that are not parallel to the feature axes since a full covariance matrix is used. The global MLE estimator should be selected if there are no significant scale variations within features and no outliers. The global MLE estimator estimates an identical bandwidth matrix for all kernels, and therefore cannot accurately model variations within features or outliers. The global MLE estimator can model correlation between features to some degree, similar to the spherical ML-LOO estimator. However, local variations of the estimated density function in the proximity of the kernels are restricted to be parallel to the feature axes.

The MLL and MLE estimators can accurately model scale variation between fea-

tures since a unique kernel bandwidth matrix is estimated for each kernel, this implies that these estimators should be able to model outliers, since they can make bandwidths of kernels centred on outliers sufficiently wide. The estimators can also model correlation between features to some degree, similar to the spherical ML-LOO and global MLE estimators, but are also restricted to estimate local scale variations parallel to the feature axes in the proximity of kernels. In our theoretical derivation, we have also noted that the MLL estimator implicitly assumes that the density function changes relatively slowly throughout feature space, therefore the MLE estimator should be preferred if some regions of the density function change quite fast.

7.3 SUMMARY OF CONTRIBUTIONS

We have introduced two new ML kernel bandwidth estimators, named the MLE and global MLE estimators, as well as a new bandwidth initialisation procedure named the MLE rule-of-thumb estimator. We have provided theoretical proofs of these estimators, and have derived the MLL and ML-LOO estimators from the same ML framework these theoretical derivations provide us with a better intuitive understanding of the behaviour of these estimators. We have shown that the contribution of the global MLE and MLE estimators are valuable since they provide researchers and practitioners with two alternative estimators to employ, which might perform optimally under certain conditions. We have also shown that the MLE and MLL estimators converge rapidly (often in only a single iteration), and therefore these estimators present two advantages of the other ML estimators investigated, namely (1) reduced training times due to fewer training iterations and (2) they may be used without an iterative bandwidth optimisation procedure e.g. if they are initialised with the Silverman rule-of-thumb, an analytical expression can be used to calculate a unique bandwidth matrix for each kernel, and no iterative bandwidth optimisation procedure will be required to update the bandwidths.

We have also gathered additional evidence that the full covariance ML-LOO and global MLE estimators are extremely powerful tools for density estimation in general pattern recognition problems these estimators often outperform the MLE and MLL estimators that have more parameters. This is an interesting case of the bias-variance trade-off: having fewer parameters, these estimators are less flexible than the MLE and MLL estimators; however, those parameters can be estimated with greater reliability, leading to the best performance in many cases. We return to this matter below.

The methodology that we have developed for using artificial data sets to understand

the strengths and weaknesses of different approaches has proven to be useful in this and other work [26]. We believe that this methodology in itself presents a useful contribution to the pattern-recognition researchers toolbox.

7.4 SUGGESTIONS FOR FURTHER RESEARCH

The full covariance ML-LOO, global MLE and MLE estimators have proven to be very powerful bandwidth estimator in this study. We noted in Section 7.2 that the global MLE estimator is identical in all respects to the full covariance ML-LOO estimator, except that it does not model correlation in the estimated bandwidth matrix. The superior performance of the full covariance ML-LOO estimator over the global MLE estimator on global PCA data shows that making use of a full covariance matrix, as opposed to a diagonal covariance matrix leads to performance improvements in practice when no class-specific transformation is performed. This is somewhat surprising, since we also explained in Section 7.2 that kernel estimators with diagonal bandwidth matrices can model correlation between features to some degree, since kernels are placed over data points and the density estimate can capture directions of variation that are not necessarily parallel to the feature axes (e.g. if data points are not aligned in direction parallel to the feature space). Diagonal kernel bandwidths do, however, restrict the local density function in the proximity of kernels to be parallel to the feature axes.

We have also noted that since the full covariance ML-LOO estimator estimates an identical bandwidth matrix for all kernels, this estimator cannot adapt the bandwidth of kernels throughout feature space to model scale within features and thus cannot adapt bandwidths to be sufficiently wide for outliers. Modelling spatial variability is, however, the strength of the MLE estimator and we observed in many cases that the MLE estimator performed optimally where the ML-LOO(f) estimator underperformed.

From these theoretical and empirical insights it is clear that the optimal estimator should somehow function like the full covariance ML-LOO estimator in regions with low spatial variability, and must function like the MLE estimator and be able to adapt bandwidths in regions with high spatial variability, especially for outliers.

We therefore believe that it would be interesting to investigate a hybrid kernel bandwidth estimator, where the correlation coefficients of the kernel bandwidth estimated by the full covariance ML-LOO estimator are used to adapt the unique diagonal kernel bandwidth matrices of each kernel. Geometrically this will imply that each kernel will have a unique kernel bandwidth that can vary in the directions parallel to the eigenvectors of the ML-LOO full covariance bandwidth matrix. This estimator

will thus be able to model more accurately scale variations between features (including outliers), and will be able to account for local variations in scale that are not parallel to the features axes, but are parallel to the eigenvectors of the ML-LOO full covariance bandwidth matrix.

For future work we propose to implement this hybrid ML kernel bandwidth estimator and perform a comparative study between the MLE, full covariance ML-LOO and the hybrid estimator.

An alternative hybrid approach with fewer parameters can also be employed by first detecting and removing outliers. Clustering can then be performed on the remaining data and the full covariance ML-LOO bandwidth estimator can be used to estimate a unique full covariance kernel bandwidth for each cluster, each kernel with thus make use the full covariance bandwidth matrix of the cluster to which it belongs. The MLE estimator can then be used to estimate a unique bandwidth for each outlier. Since the MLE estimator can model scale variations, this will ensure that outliers have sufficiently wide bandwidths. This proposed hybrid approach will thus lead to less variance than the first hybrid approach, since fewer bandwidth parameters are estimated.

We therefore propose the implement this hybrid ML kernel bandwidth estimator in future work and perform a comparative study between this approach, the MLE, full covariance ML-LOO and the first hybrid approach proposed above.

We have highlighted some of the effects of bandwidth initialisation on the performance of ML estimators in this study. We noted that the bandwidth to which the ML bandwidth estimators converge is dependent on the initial bandwidth and therefore the ideal bandwidth initialisation algorithm should initialise bandwidths close to global minima. Our comparative investigation of the MLE rule-of-thumb and Silverman rule-of-thumb estimators for bandwidth initialisation showed that neither of these estimators were optimal for all estimators on all tasks, and therefore the selection of the ideal bandwidth initialisation estimator will probably be dependent on the properties of the data and the properties of the ML bandwidth estimator being initialised. We propose to perform a comparative study of bandwidth initialisation algorithms in the future, to gain a better understanding of the behaviour of the ML bandwidth estimators for different bandwidth initialisation algorithms on different tasks.

In future work we will also like to extend our work to unsupervised learning by performing mean-shift clustering that makes use of the ML KDEs proposed and investigated in this study. The mean-shift clustering algorithm requires the estimation of a pilot density and data points are clustered by assigning each data point to the nearest mode of the underlying density function. Each data point is assigned to a mode by

iteratively moving the data point in the direction of most density, therefore the estimation of the underlying density function is crucial to the success of the mean-shift clustering algorithm.

We will also like to extend our work to supervised learning by performing model-based classification on class-conditional PDFs estimated with the ML KDEs proposed and investigated in this work. We note that there are some challenges with performing model-based Bayes classification, most notably that the same transformation needs to be applied to all classes, in order to compare the likelihood scores between classes. Therefore, several innovations will be required to perform model-based classification on data that is transformed with class-specific transformations, since the likelihoods obtained from class-conditional PDFs that are estimated from class-specific PCA data are not directly comparable between classes.

7.5 CONCLUSION

This research has shown that kernel density estimation with ML kernel bandwidth estimation is a suitable candidate for non-parametric density estimation on real-world high-dimensional pattern-recognition tasks. This study also showed that conventional density estimators, such as the Silverman rule-of-thumb, can also be used for kernel density estimation on real-world high-dimensional pattern-recognition tasks with some degree of success. The ML kernel bandwidth estimators proposed and derived in this study do, however, give significant performance improvements over the conventional density estimators investigated in this study.

We simulated artificial data sets with data properties that were varied in a controlled fashion to show the strengths and weaknesses of the ML bandwidth estimators investigated. The theoretical insights obtained from our derivation of the ML bandwidth estimators guided us in selecting the appropriate data properties to simulate and vary.

The results of our comparative simulation studies showed that no conventional or ML kernel bandwidth estimator performed optimally on all tasks, and that the selection of the appropriate bandwidth estimator should rather be based on the properties of the data. The MLE and full covariance ML-LOO estimators did, however, show much promise and we therefore proposed two potential hybrid approaches based on these estimators to perform kernel bandwidth estimation, in the quest to find a kernel bandwidth estimator that performs optimally on all types of data.

Kernel density estimation has great theoretical interest and practical importance

and we hope that our theoretical contributions of the MLE and global MLE ML kernel bandwidth estimators and the results of our empirical studies will lead to further progress in the field of pattern recognition.

References

[1] Oystein Ore, "Pascal and the invention of probability theory," *The American Mathematical Monthly*, vol. 67, no. 5, pp. 409–419, 1960.

[2] Evelyn Fix and Joseph Lawson Hodges, "Discriminatory analysis, nonparametric estimation: consistency properties," Tech. Rep. Report No. 4, Project No. 21-49-004, USAF School of Aviation Medicine, Randolph Filed, Texas, February 1951.

[3] Bernard W Silverman, *Density estimation for statistics and data analysis*, Chapman and Hall, 1986.

[4] Emanuel Parzen, "On estimation of a probability density function and mode," *The annals of mathematical statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[5] Murray Rosenblatt, "Remarks on some nonparametric estimates of a density function," *The Annals of Mathematical Statistics*, pp. 832–837, 1956.

[6] Peter Whittle, "On the smoothing of probability density functions," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 334–343, 1958.

[7] Don O Loftsgaarden and Charles P Quesenberry, "A nonparametric estimate of a multivariate density function," *The Annals of Mathematical Statistics*, pp. 1049–1051, 1965.

[8] Leo Breiman, William Meisel, and Edward Purcell, "Variable kernel estimates of multivariate densities," *Technometrics*, vol. 19, no. 2, pp. 135–144, 1977.

[9] Jerome H Friedman, Werner Stuetzle, and Anne Schroeder, "Projection pursuit density estimation," *Journal of the American Statistical Association*, vol. 79, no. 387, pp. 599–608, 1984.

[10] M Chris Jones and Robin Sibson, "What is projection pursuit?," *Journal of the Royal Statistical Society. Series A (General)*, pp. 1–37, 1987.

[11] Arthur P Dempster, Nan M Laird, and Donald B Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 1–38, 1977.

[12] Richard A Redner and Homer F Walker, "Mixture densities, maximum likelihood and the em algorithm," *SIAM review*, vol. 26, no. 2, pp. 195–239, 1984.

[13] Jeff A Bilmes et al., "A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models," *International Computer Science Institute*, vol. 4, no. 510, pp. 126, 1998.

[14] Andrew R. Webb, *Statistical Pattern Recognition*, John Wiley & Sons, 2002.

[15] David Heckerman, *A tutorial on learning with Bayesian networks*, Springer, 2008.

[16] David W Scott and Stephan R Sain, "Multi-dimensional density estimation," *Handbook of Statistics*, vol. 24, no. 2004, pp. 229–261, 2005.

[17] Etienne Barnard, "Maximum leave-one-out likelihood for kernel density estimation," *Proceedings of the twenty-first annual symposium of the Pattern Recognition Association of South Africa (PRASA)*, November 2010.

[18] José M Leiva-Murillo and Antonio Artés-Rodríguez, "Algorithms for maximum-likelihood bandwidth selection in kernel density estimators," *Pattern Recognition Letters*, vol. 33, no. 13, pp. 1717–1724, 2012.

[19] Ana Justel, Daniel Peña, and Rubén Zamar, "A multivariate kolmogorov-smirnov test of goodness of fit," *Statistics & Probability Letters*, vol. 35, no. 3, pp. 251–259, 1997.

[20] Herbert A Sturges, "The choice of a class interval," *Journal of the American Statistical Association*, vol. 21, no. 153, pp. 65–66, 1926.

[21] David W Scott, "Frequency polygons: theory and application," *Journal of the American Statistical Association*, vol. 80, no. 390, pp. 348–354, 1985.

[22] David W Scott, "Averaged shifted histograms: effective nonparametric density estimators in several dimensions," *The Annals of Statistics*, vol. 13, no. 3, pp. 1024–1040, 1985.

[23] George R Terrell and David W Scott, "Variable kernel density estimation," *The Annals of Statistics*, pp. 1236–1265, 1992.

[24] Dimitrios Ververidis and Constantine Kotropoulos, "Gaussian mixture modeling by exploiting the mahalanobis distance," *Signal Processing, IEEE Transactions on*, vol. 56, no. 7, pp. 2797–2811, 2008.

[25] David W Scott and William F Szewczyk, "From kernels to mixtures," *Technometrics*, vol. 43, no. 3, pp. 323–335, 2001.

[26] Christiaan M. van der Walt, "Data measures that characterise classification problems," 2008, Masters Dissertation.

[27] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei, "Hierarchical dirichlet processes," *Journal of the american statistical association*, vol. 101, no. 476, 2006.

[28] David M Blei and Michael I Jordan, "Variational methods for the dirichlet process," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 12.

[29] Carl Edward Rasmussen, "The infinite gaussian mixture model.," in *NIPS*, 1999, vol. 12, pp. 554–560.

[30] Wray Buntine, "A guide to the literature on learning probabilistic networks from data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 8, no. 2, pp. 195–210, 1996.

[31] Jorma Rissanen, "A universal prior for integers and estimation by minimum description length," *The Annals of statistics*, pp. 416–431, 1983.

[32] Claude Elwood Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[33] Alan Julian Izenman, "Review papers: Recent developments in nonparametric density estimation," *Journal of the American Statistical Association*, vol. 86, no. 413, pp. 205–224, 1991.

[34] Berwin A Turlach, "Bandwidth selection in kernel density estimation: A review," *CORE and Institut de Statistique*, vol. 19, no. 4, pp. 1–33, 1993.

[35] Simon J Sheather, "Density estimation," *Statistical Science*, pp. 588–597, 2004.

[36] M Chris Jones, James S Marron, and Simon J Sheather, "A brief survey of bandwidth selection for density estimation," *Journal of the American Statistical Association*, vol. 91, no. 433, pp. 401–407, 1996.

[37] Mats Rudemo, "Empirical choice of histograms and kernel density estimators," *Scandinavian Journal of Statistics*, pp. 65–78, 1982.

[38] Simon J Sheather and Michael C Jones, "A reliable data-based bandwidth selection method for kernel density estimation," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 683–690, 1991.

[39] J.H. Zar, *Density estimation for statistics and data analysis*, Chapman and Hall, 1986.

[40] A. Trujillo-Ortiz and R. Hernandez-Walls, "skekurtest: Hypotheses test concerning skewness," `http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=3953&objectType=FILE`, 2003.

[41] NIST, "Nist/sematech e-handbook of statistical methods: Critical values of the student's t distribution," `http://www.itl.nist.gov/div898/handbook/eda/section3/eda3672.htm`, 2012.

[42] K. Bache and M. Lichman, "UCI machine learning repository," `http://archive.ics.uci.edu/ml`, 2013.

[43] David W Scott and James R Thompson, "Probability density estimation in higher dimensions," in *Computer Science and Statistics: Proceedings of the Fifteenth Symposium on the Interface*. North-Holland, Amsterdam, 1983, vol. 528, pp. 173–179.

[44] Luis O Jimenez and David A Landgrebe, "Supervised classification in high-dimensional space: geometrical, statistical, and asymptotical properties of multivariate data," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 28, no. 1, pp. 39–54, 1998.

[45] Jörg Bruske and Gerald Sommer, "Intrinsic dimensionality estimation with optimally topology preserving maps," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 5, pp. 572–575, 1998.

[46] Ann B Lee, Kim S Pedersen, and David Mumford, "The nonlinear statistics of high-contrast patches in natural images," *International Journal of Computer Vision*, vol. 54, no. 1-3, pp. 83–103, 2003.

[47] Etienne Barnard, Christiaan M. van der Walt, Marelie Davel, Charl van Heerden, Fred Senekal, and Thegaran Naidoo, "Learning structured representations of data," in *Proceedings of the twentieth annual symposium of the Pattern Recognition Association of South Africa (PRASA)*, Stellenbosch, South Africa, November 2009.

[48] Etienne Barnard, "Visualizing data in high-dimensional spaces," in *Proceedings of the twenty-first annual symposium of the Pattern Recognition Association of South Africa (PRASA)*, Stellenbosch, South Africa, November 2010.

[49] LJP Van der Maaten, EO Postma, and HJ Van Den Herik, "Dimensionality reduction: A comparative review," *Journal of Machine Learning Research*, vol. 10, pp. 1–41, 2009.

[50] Kaare Brandt Petersen and Michael Syskind Pedersen, "The matrix cookbook," 2006.

[51] Ashis Gangopadhyay and Kin Cheung, "Bayesian approach to the choice of smoothing parameter in kernel density estimation," *Journal of Nonparametric Statistics*, vol. 14, no. 6, pp. 655–664, 2002.

[52] Christiaan M. Van Der Walt and Etienne Barnard, "Density estimation from local structure," in *Proceedings of the twentieth annual symposium of the Pattern Recognition Association of South Africa (PRASA)*, Stellenbosch, South Africa, November 2009.

[53] A. Ihler and M. Mandel, "Kernel density estimation toolbox for matlab," `http://www.ics.uci.edu/ihler/code/`, 2003.

Appendix A

Conventional Estimator Results (Global PCA)

Contents

A.1 Introduction 172

A.2 CV Global PCA Results 173

A.3 RW Global PCA Results 175

A.4 Conclusion 180

A.1 INTRODUCTION

In Chapter 5 we presented the comparative entropy results of the MLE (init), and the conventional ULCV, LLCV, LSCV, Silverman, MSP and ML Gauss estimators for class-specific PCA5 and PCA1 datasets. In this appendix, we present similar comparative results for data pre-processed with the global PCA transformation (i.e. where the data of all classes are pre-processed simultaneously and not separately).

The results for the global PCA5 and PCA1 CV datasets are be presented in Section A.2, the results for the global PCA5 and PCA1 RW datasets are presented in Section A.3, and we make concluding remarks regarding the difference in estimator performance for class-specific and globally pre-processed data in Section A.4.

A.2 CV GLOBAL PCA RESULTS

Table A.1: CV entropy results (global PCA5)

DS	C	K	MLE(i)	ULCV(s)	LLCV(s)	LSCV(s)	Silverman	MSP	HSJM	Gauss
OF	1	2	2.1619	1.5114	1.5190	1.6933	1.6418	1.7041	1.5120	2.0179
BS	1	4	5.5255	5.3827	5.7912	5.4015	5.0586	5.0803	5.3441	5.1328
	2	4	5.8723	6.1056	6.3945	6.1056	5.3802	5.4675	5.6115	4.2340
	3	4	5.5156	5.3312	5.7345	5.3924	5.0530	5.0735	5.3489	5.1190
IR	1	4	0.4097	0.6097	0.8339	0.6160	0.6054	0.4742	2.0713	0.3856
	2	4	1.6872	3.0288	2.9977	3.0288	1.3272	1.3701	1.7273	1.1959
	3	4	2.6355	2.6899	2.7264	2.7189	2.3406	2.3654	2.8414	2.2284
DB	1	5	7.0784	6.6440	7.8185	6.6322	6.6602	6.6504	8.1268	7.1364
	2	5	8.0751	7.6870	7.8621	7.6890	7.7061	7.7200	8.6654	7.9521
HS	1	5	8.0708	7.9029	7.9119	7.8905	7.9363	7.9079	8.9801	7.8740
	2	5	7.9269	7.7098	8.1904	7.7278	7.7300	7.7216	8.6364	7.7928
VC	1	5	8.4126	8.3095	8.0612	8.3095	7.7023	7.8448	7.7250	8.1341
	2	5	8.2524	8.5665	8.2942	8.5665	7.5803	7.6898	7.6911	7.9314
	3	5	8.2543	7.2191	11.1752	7.3468	7.2683	7.3624	8.2200	8.8648
	4	5	8.2262	7.4560	7.3807	7.4780	7.0155	7.0756	7.6642	8.4850
WF	1	5	7.9968	7.7105	7.8827	7.8704	7.7245	7.7105	8.2330	7.6424
	2	5	8.0400	7.4660	7.6244	7.6038	7.4709	7.5056	7.9823	7.3162
	3	5	8.0889	7.4788	7.4880	7.6200	7.4900	7.5339	7.9882	7.3367
IS	1	5	10.1883	10.0472	10.0877	15.1409	10.4351	10.2326	14.7161	10.1244
	2	5	8.2399	6.5278	6.6624	6.6534	6.7033	6.7698	10.2014	8.7116

Table A.2: CV entropy results (global PCA1)

DS	C	K	MLE(i)	ULCV(s)	LLCV(s)	LSCV(s)	Silverman	MSP	HSJM	Gauss
OF	1	2	2.1619	1.5114	1.5190	1.6933	1.6418	1.7041	1.5120	2.0179
BS	1	4	5.5255	5.3827	5.7912	5.4015	5.0586	5.0803	5.3441	5.1328
	2	4	5.8723	6.1056	6.3945	6.1056	5.3802	5.4675	5.6115	4.2340
	3	4	5.5156	5.3312	5.7345	5.3924	5.0530	5.0735	5.3489	5.1190
IR	1	3	1.1919	1.1504	1.6419	1.2447	1.2482	1.1872	2.3054	1.1224
	2	3	2.3819	2.0399	2.1092	2.2753	2.0380	2.0891	2.3086	1.9404
	3	3	2.8695	2.6038	2.5946	2.7340	2.5585	2.5912	3.0824	2.5816
DB	1	8	9.6896	9.5820	9.1986	9.5820	9.2695	9.1916	10.5263	10.0603
	2	8	11.3546	11.1065	11.0432	11.1065	10.9108	10.9148	11.5358	11.5527
HS	1	13	18.0994	21.1561	21.8631	21.1561	18.4087	18.1940	18.2825	17.9901
	2	13	16.5999	17.0466	19.7891	20.4335	16.4582	16.5064	16.4933	16.9467
VC	1	9	10.9098	14.5636	13.7379	14.5636	9.8325	10.1829	9.6362	9.9579
	2	9	10.5295	15.5109	14.8863	15.5109	9.4816	9.7974	9.4029	9.5411
	3	9	10.6489	11.3088	21.0048	11.3088	9.2641	9.5578	9.8224	10.9718
	4	9	10.7090	13.5783	13.6598	13.5783	9.3178	9.4997	9.8865	11.2336
WF	1	21	24.7860	43.2885	41.7748	43.2885	25.3210	24.9533	35.5649	23.1161
	2	21	24.8560	43.6120	44.5429	43.6120	25.2677	24.9739	34.3139	22.8894
	3	21	24.9922	44.1091	47.0836	44.1091	25.4349	25.1159	35.4929	22.9442
IS	1	32	49.3361	55.1791	54.2179	55.3489	53.1453	50.6898	67.0522	57.7260
	2	32	4.1282	16.0464	19.9399	16.0464	8.0738	6.2969	22.7313	8.4025

A.3 RW GLOBAL PCA RESULTS

Table A.3: Letter test entropy results (global PCA5)

C	K	NC	MLE(i)	ULCV(s)	LLCV(s)	LSCV(s)	Silverman	MSP	HSJM	Gauss
1	5	789	6.1498	4.1064	4.3868	4.0940	4.4807	4.7038	4.6905	5.2942
2	5	766	6.2292	4.4555	4.7257	4.4589	4.8409	5.0366	4.9525	6.0746
3	5	736	6.7996	4.7643	4.8396	4.7561	5.3525	5.6044	4.9714	6.2851
4	5	805	6.2742	4.3556	4.3987	4.3504	4.7560	4.9642	4.6678	5.9210
5	5	768	6.9594	4.4097	4.6408	4.4231	5.2943	5.5828	4.4801	6.7013
6	5	775	7.2440	4.9457	5.2264	4.9450	5.6877	5.9501	5.2401	6.7902
7	5	773	6.5573	4.7071	4.8739	4.7075	5.2665	5.5054	5.0942	6.3984
8	5	734	7.1654	4.6567	5.1964	4.6567	5.4588	5.7403	5.0005	6.9831
9	5	755	6.8758	3.6115	4.0422	3.6368	5.0464	5.4137	3.7013	6.9592
10	5	747	7.7839	5.0126	5.5566	5.0230	6.0874	6.3818	4.9640	7.1299
11	5	739	7.6884	4.9167	5.4839	4.9212	5.9215	6.2263	5.0231	7.3387
12	5	761	6.7813	4.5921	4.8841	4.6277	5.2227	5.4832	4.8724	6.3389
13	5	792	7.8202	5.0740	5.3804	5.1295	6.1587	6.4619	5.1562	7.5492
14	5	783	7.7416	4.7042	5.3528	4.8006	5.8310	6.1458	4.5372	7.4804
15	5	753	5.4484	4.2068	4.2631	4.2049	4.4154	4.5763	4.6840	5.3744
16	5	803	6.8005	5.1427	5.3780	5.1501	5.5873	5.7958	5.3732	6.6207
17	5	783	6.2659	5.1162	5.1685	5.1162	5.1108	5.2456	5.9776	6.0837
18	5	758	6.2570	4.2855	4.5201	4.3040	4.8538	5.0920	4.6190	5.6479
19	5	748	7.1526	4.4674	4.4238	4.4887	5.5000	5.8003	4.4624	6.9611
20	5	796	6.9132	4.5730	4.6516	4.5737	5.3380	5.6100	4.7902	6.6295
21	5	813	6.9531	4.2899	4.6156	4.3285	5.3533	5.6659	4.4274	6.1887
22	5	764	6.0499	3.9004	4.2525	3.9034	4.6480	4.9201	4.0948	5.4878
23	5	752	7.0389	5.1723	5.2026	5.1631	5.6260	5.8505	5.4747	6.9459
24	5	787	6.9244	4.4769	5.0570	4.5596	5.3235	5.6127	4.5449	6.8430
25	5	786	7.3411	5.0071	5.3345	5.0071	5.8097	6.0933	5.1580	6.8003
26	5	734	7.1631	4.6547	4.7435	4.7241	5.6337	5.9365	4.6770	6.8704

Table A.4: Letter test entropy results (global PCA1)

C	K	NC	MLE(i)	ULCV(s)	LLCV(s)	LSCV(s)	Silverman	MSP	HSJM	Gauss
1	16	789	11.3753	15.6692	14.9798	15.6692	7.8747	9.0246	5.8065	10.4935
2	16	766	10.2394	12.9995	13.4936	12.9995	7.5631	8.5807	6.2579	9.3682
3	16	736	10.9874	17.3584	15.9312	17.3584	8.2122	9.3968	6.1344	9.8206
4	16	805	10.2153	17.0530	15.4219	17.0530	7.0506	8.1016	5.4732	9.6923
5	16	768	10.8532	14.4589	15.5591	14.4589	7.9685	9.2030	5.4085	9.5728
6	16	775	11.6353	15.8793	18.4380	15.8793	8.6868	9.9453	5.8131	9.7828
7	16	773	10.9785	17.8397	16.7626	17.8397	8.4623	9.4983	7.0052	10.5565
8	16	734	12.5717	18.4147	18.7142	18.4147	9.5594	10.7406	7.3730	12.2497
9	16	755	10.4617	13.8791	14.2682	13.8791	7.0100	8.2681	4.5387	10.2531
10	16	747	12.8932	18.1627	18.4949	18.1627	9.6965	11.0388	6.2228	10.7591
11	16	739	12.9629	19.0078	19.7162	19.0078	10.1542	11.3248	8.1946	11.8430
12	16	761	13.5193	18.7324	21.6551	18.7324	10.1923	11.5270	6.8285	10.8425
13	16	792	13.9988	17.6139	16.8868	17.6139	10.6838	11.9781	7.5545	13.3691
14	16	783	12.9034	15.2040	17.1869	15.2040	9.4188	10.6648	6.5658	12.2999
15	16	753	8.7092	13.9967	14.3050	13.9967	6.1856	7.0848	5.1879	8.7089
16	16	803	12.0619	15.7966	15.1072	15.7966	9.2493	10.4548	6.8437	10.7307
17	16	783	11.8239	17.0519	15.0905	17.0519	8.9068	10.0102	7.0638	11.2537
18	16	758	11.1283	14.2565	15.1711	14.2565	8.4048	9.5836	6.3970	9.8374
19	16	748	11.9392	21.7858	22.8846	21.7858	8.8927	10.1373	6.6633	10.7833
20	16	796	12.1750	18.9012	20.5288	18.9012	9.3176	10.6618	5.5912	10.0264
21	16	813	12.8726	19.9928	22.0630	19.9928	10.0412	11.3864	6.3094	11.3174
22	16	764	11.2419	13.4049	14.9466	13.4049	8.5769	9.7894	5.9781	9.6703
23	16	752	12.3585	12.5989	14.0896	12.5989	9.5264	10.7568	7.1849	11.6283
24	16	787	10.6294	13.4397	13.0390	13.4397	8.0153	9.1819	6.1487	10.2765
25	16	786	13.9051	14.2886	13.5259	14.2886	10.7121	12.0284	7.4923	12.6734
26	16	734	11.0935	14.3976	13.3042	14.3976	7.9734	9.0731	6.4490	10.9005

Table A.5: Segmentation test entropy results (global PCA5)

C	K	NC	MLE(i)	ULCV(s)	LLCV(s)	LSCV(s)	Silverman	MSP	HSJM	Gauss
1	5	330	4.0340	2.9175	2.8849	2.4596	2.8943	3.0760	2.7400	3.8737
2	5	330	4.6505	3.7075	3.3986	4.3252	3.6381	3.8354	4.5699	4.5099
3	5	330	5.2272	2.4087	2.0866	5.3730	3.2260	3.5407	2.5943	5.4168
4	5	330	5.7606	4.1997	4.6464	4.4912	4.4665	4.7016	4.4187	5.9348
5	5	330	7.1597	7.8546	19.4183	24.5875	10.5345	9.2133	25.5708	7.7506
6	5	330	2.5603	0.5759	0.0473	0.0604	1.2509	1.5327	0.2334	2.3545
7	5	330	3.8162	0.4341	0.5844	0.0614	1.9749	2.3639	-0.0153	3.0062

Table A.6: Segmentation test entropy results (global PCA1)

C	K	NC	MLE(i)	ULCV(s)	LLCV(s)	LSCV(s)	Silverman	MSP	HSJM	Gauss
1	13	330	3.9872	3.7943	4.6353	3.7943	1.1732	1.9836	0.0555	2.9833
2	13	330	6.7069	5.6263	9.2153	5.6263	1.6428	2.4973	1.4855	0.3542
3	13	330	7.7302	2.4249	2.8152	41.7236	4.3784	5.0890	9.0779	7.1357
4	13	330	9.8721	10.1214	10.9195	10.1214	13.5635	12.5517	8.8971	10.7352
5	13	330	14.6143	16.2057	40.0529	30.8928	33.9773	27.5999	49.8414	15.7465
6	13	330	3.0450	0.6104	-0.8392	0.6104	0.0030	1.1641	-4.2616	-7.5550
7	13	330	-0.0452	2.3311	6.1404	2.3311	-3.5933	-2.3950	-7.8366	-6.1529

Table A.7: Landsat test entropy results (global PCA5)

C	K	NC	MLE(i)	ULCV(s)	LLCV(s)	LSCV(s)	Silverman	MSP	HSJM	Gauss
1	5	1533	6.2102	4.9099	5.1117	4.9341	5.1065	5.2262	6.2531	5.7720
2	5	703	8.0346	5.9907	6.3507	6.0880	6.4365	6.6699	7.0110	7.6676
3	5	1358	4.8215	3.6802	6.2167	3.7053	3.7548	3.7882	5.9507	4.7301
4	5	626	5.1791	4.0797	5.3616	4.0377	4.1524	4.2286	5.5860	5.1979
5	5	707	7.7632	6.5923	6.8530	6.5923	6.7684	6.9147	7.3250	7.6813
6	5	1508	5.2712	4.1213	7.6954	4.1288	4.2341	4.3301	6.1242	5.1635

Table A.8: Landsat test entropy results (global PCA1)

C	K	NC	MLE(i)	ULCV(s)	LLCV(s)	LSCV(s)	Silverman	MSP	HSJM	Gauss
1	8	1533	7.7838	7.2893	6.6939	7.2893	6.6706	6.7556	9.3083	7.5483
2	8	703	11.2533	10.5407	9.7608	10.5407	8.9160	9.3267	10.1519	10.8913
3	8	1358	6.1882	5.5664	11.6865	5.5664	5.0805	5.0445	7.9326	6.1619
4	8	626	6.6553	6.4194	12.7370	6.4194	5.3579	5.3912	7.4971	7.0353
5	8	707	10.3179	9.8178	11.2083	9.8178	9.1022	9.2759	10.3406	10.5415
6	8	1508	6.6896	5.4840	13.7740	5.4840	5.5442	5.5770	9.4711	6.8484

Table A.9: Optdigits test entropy results (global PCA5)

C	K	NC	MLE(i)	ULCV(s)	LLCV(s)	LSCV(s)	Silverman	MSP	HSJM	Gauss
1	5	554	6.3192	5.9477	7.6553	5.9484	5.9822	5.9460	7.3914	6.0264
2	5	571	9.0498	7.4581	7.7544	7.4478	7.6059	7.7221	8.6235	9.0345
3	5	557	8.6072	8.0880	8.2234	8.0880	8.0652	8.0632	10.1366	8.5774
4	5	572	7.9542	7.5846	9.4451	7.5776	7.6286	7.5781	9.3064	7.9401
5	5	568	9.5142	8.5888	9.8001	8.5940	8.6468	8.7775	9.3918	9.0481
6	5	558	8.7944	8.0489	8.0437	8.0290	8.0347	8.0867	9.2676	8.5004
7	5	558	7.5305	6.8448	8.2073	6.8485	6.8414	6.8952	8.0015	6.9048
8	5	566	8.1911	7.5605	8.7557	7.5605	7.4467	7.4933	9.5622	7.9468
9	5	554	8.1172	7.8436	7.8294	7.8228	7.8588	7.8193	8.9829	7.7942
10	5	562	8.5331	7.6750	7.6708	7.6752	7.7433	7.8630	8.6066	8.3154

Table A.10: Optdigits test entropy results (global PCA1)

C	K	NC	MLE(i)	ULCV(s)	LLCV(s)	LSCV(s)	Silverman	MSP	HSJM	Gauss
1	60	554	32.8501	125.6073	142.4578	125.6073	32.2529	34.1598	115.4551	46.9870
2	60	571	45.6307	139.6125	166.3361	139.6125	41.5517	43.4502	73.9364	71.0936
3	60	557	55.5458	172.8604	195.5486	172.8604	53.4374	54.7818	112.9349	73.5198
4	60	572	48.5282	141.6640	183.7962	141.6640	50.8373	50.8819	128.7071	58.9548
5	60	568	60.5200	174.4953	168.6387	174.4953	59.3826	61.0577	105.7475	76.2713
6	60	558	53.9097	148.2743	148.4916	148.2743	52.4564	53.4266	118.2641	64.4287
7	60	558	40.2889	175.0980	199.4358	175.0980	38.9048	41.5438	100.7760	61.7806
8	60	566	52.1771	170.1742	175.1781	170.1742	49.9243	52.0542	105.4759	67.1796
9	60	554	52.4554	159.8908	164.1701	159.8908	54.6098	54.5826	96.0278	62.3070
10	60	562	50.9729	150.6896	165.0364	150.6896	51.4175	52.4227	135.9377	61.7475

Table A.11: Isolet test entropy results (global PCA5)

C	K	NC	MLE(i)	ULCV(s)	LLCV(s)	LSCV(s)	Silverman	MSP	HSJM	Gauss
1	5	300	13.6826	13.1761	13.5548	13.1364	13.1284	13.1881	13.9180	13.1435
2	5	300	13.5864	12.9340	13.6223	12.9440	12.9724	13.0661	13.5205	12.6873
3	5	300	13.6581	13.0532	13.0756	13.1722	13.0650	13.1372	13.5378	12.9690
4	5	300	13.8139	13.1848	13.6858	13.1428	13.1473	13.2301	13.9431	13.0933
5	5	300	13.7466	13.3967	13.3714	13.2993	13.3030	13.3360	14.3865	13.1159
6	5	300	12.9329	12.6161	12.7756	12.6161	12.5353	12.5767	13.3058	12.3290
7	5	300	13.7066	12.6403	12.6031	12.7522	12.7704	12.9294	12.8696	12.6422
8	5	300	13.6086	12.8378	12.9983	12.9804	12.8956	13.0195	13.0502	12.8903
9	5	300	13.2040	12.9747	12.9196	12.9254	12.9402	12.9078	14.0511	12.6381
10	5	300	13.3962	12.5610	12.8437	12.6424	12.6352	12.7522	13.0166	12.6210
11	5	300	13.2970	12.3166	12.3179	12.3987	12.4507	12.5976	12.7647	12.4801
12	5	300	13.2306	12.8426	12.8410	12.8856	12.7974	12.8295	13.4235	12.8327
13	5	300	13.3959	13.0219	13.2182	13.0555	13.0223	13.0536	13.7909	12.8847
14	5	300	13.1888	12.8383	13.0280	12.8664	12.8364	12.8638	13.5921	12.5203
15	5	300	14.0114	13.4727	13.4729	13.5936	13.4837	13.5540	13.8097	13.3002
16	5	300	13.7617	12.7350	12.8675	12.8556	12.8926	13.0451	13.1415	12.9142
17	5	300	13.4015	12.8616	12.9029	12.9580	12.8922	12.9725	13.1855	12.6206
18	5	300	13.1691	12.7379	12.8285	12.7745	12.7024	12.7281	13.2797	12.4501
19	5	300	12.8291	12.6223	12.6264	12.6090	12.6011	12.5886	13.6206	12.3023
20	5	300	13.8776	13.1097	13.1411	13.0797	13.1377	13.2448	13.9112	13.3980
21	5	300	13.3528	13.0094	13.0632	13.1903	13.0068	13.0738	13.2708	12.8313
22	5	300	14.0808	13.8112	13.8977	13.7592	13.7915	13.7922	14.8014	13.5000
23	5	300	14.8005	14.4976	14.7081	14.5508	14.5052	14.5081	15.1903	14.4493
24	5	300	12.7981	12.5008	12.5170	12.4856	12.4785	12.4836	13.8555	12.3098
25	5	300	12.5823	12.1508	12.2643	12.1785	12.1508	12.1828	12.8613	11.7191
26	5	300	14.0726	14.0239	14.0040	13.8603	13.9536	13.8631	15.2133	13.6283

Table A.12: Isolet test entropy results (global PCA1)

C	K	NC	MLE(i)	ULCV(s)	LLCV(s)	LSCV(s)	Silverman	MSP	HSJM	Gauss
1	64	300	123.1327	237.0268	251.7373	237.0268	124.7601	123.8568	264.5351	112.9564
2	64	300	122.8669	229.2356	266.5634	229.2356	123.7344	123.9629	249.3259	112.9246
3	64	300	123.9538	228.8303	239.1685	228.8303	125.8186	126.0061	229.6165	114.8896
4	64	300	123.8000	224.7701	263.2216	224.7701	125.3082	124.6732	204.7872	115.9373
5	64	300	126.5324	224.0191	229.0790	224.0191	127.8184	127.4056	200.7209	119.7994
6	64	300	123.3641	209.4881	213.8762	209.4881	124.3480	124.3639	221.4123	116.1408
7	64	300	121.8413	237.4826	243.0240	237.4826	122.8798	122.3443	267.1798	111.5281
8	64	300	127.5440	230.1955	246.2644	230.1955	128.2589	128.4186	225.2097	112.7244
9	64	300	126.1583	237.0714	234.3474	237.0714	127.3955	126.5686	296.4544	116.8452
10	64	300	120.8410	198.3507	206.1101	198.3507	121.3175	120.8250	180.1479	121.1442
11	64	300	119.8680	220.5076	216.1074	220.5076	120.8827	120.3111	248.4044	115.3458
12	64	300	126.1216	241.9755	249.1572	241.9755	127.1991	126.8980	270.9614	117.2820
13	64	300	133.1551	230.0215	231.6339	230.0215	134.3412	133.5393	221.3869	124.1159
14	64	300	131.9258	230.7119	227.1286	230.7119	133.2158	132.6251	238.0965	122.2879
15	64	300	126.7855	230.9099	228.3920	230.9099	127.7360	127.5378	268.5666	111.0721
16	64	300	123.4145	237.8886	256.2286	237.8886	124.2682	124.4464	201.1682	116.7388
17	64	300	125.3721	228.5423	229.7811	228.5423	127.1580	127.2650	233.0865	113.0233
18	64	300	127.1201	231.5271	226.6796	231.5271	127.5601	127.0100	239.3960	114.7173
19	64	300	127.3590	223.1497	231.4997	223.1497	128.0174	127.9805	230.4330	118.7226
20	64	300	123.5392	207.8140	204.2328	207.8140	124.0755	124.0673	233.6703	114.8415
21	64	300	130.6404	240.6090	240.4045	240.6090	132.1341	131.9790	258.2710	120.9178
22	64	300	129.5498	236.0287	237.1053	236.0287	131.3154	130.0442	264.6039	124.7434
23	64	300	140.6231	241.7470	243.3339	241.7470	139.8064	141.2052	218.8068	132.9026
24	64	300	127.5683	243.2129	251.5641	243.2129	127.8106	127.8314	238.7809	114.3150
25	64	300	127.7553	233.8843	243.3128	233.8843	128.3753	128.5783	251.9680	114.8254
26	64	300	131.7804	219.8858	219.4519	219.8858	132.7288	132.3935	215.9656	123.6958

A.4 CONCLUSION

In Chapter 5 we concluded that for class-specific PCA data, the conventional ULCV, LLCV and LSCV estimators performed competitively on low-dimensional PCA5 data, and were mostly not suited for PCA1 datasets with dimensionalities larger than 10. The performances of the Silverman and MSP estimators were very consistent across all dimensionalities and tasks, and were generally very competitive; and the MLE(i) and ML Gauss estimators experienced relative performance increases with increased dimensionality, and might be favoured in higher dimensions that were experimented with. We also noted that the HSJM estimator performance was more difficult to predict.

The CV and RW global PCA results presented in Section A.2 and Section A.3 show that similar relative performance behaviour holds for all except the LSCV estimator. For the PCA5 Letter, Landsat and Optdigits datasets, the LSCV estimator performs more competitively than for the class-specific PCA5 data. However, this relative improvement in performance of the LSCV estimator for global PCA5 data, does not hold for the higher-dimensional PCA1 data – which confirms our observation that the LSCV is mostly not suited for datasets with high dimensionalities. We also note that the actual values of entropy scores are not comparable between datasets with different transformations, since the scaling of the transformed features are different for different transformations, owing to the different eigenvalues of the PCA transformations. The entropy of a Gaussian distribution, for example, is related to the determinant of the covariance matrix, therefore a higher entropy value will be obtained if the feature axes are scaled to have the same variance. This illustrates how different scaling of features results in non-comparable entropy values.

In summary, the relative performances of the conventional and MLE(i) estimators on the global PCA data are not markedly different from the class-specific estimator performances - with the exception of the LSCV estimator on PCA5 data.

Appendix B

Comparison of MLE and Silverman Initialisation

Contents

B.1 Introduction 181

B.2 Comparison of MLE rule-of-thumb and Silverman rule-of-thumb Initialisation 182

B.2.1 CV dataset results 182

B.2.2 RW dataset results 186

B.3 Conclusion 190

B.1 INTRODUCTION

In Section 6.2.6 we discussed the selection of an appropriate bandwidth initialisation procedure for the global MLE, MLE, MLL and ML-LOO ML estimators. We noted that the consistent performance of the Silverman rule-of-thumb estimator does not necessarily imply that the Silverman estimator will perform optimally as the default bandwidth initialisation procedure, since the bandwidths to which the ML estimators converge are dependent on their initialisation, and are thus susceptible to local minima. We motivated that the MLE rule-of-thumb estimator might be a good alternative, since

it is derived from the same framework as the ML estimators and it guarantees that bandwidths are within the bounds of the optimal ML bandwidths.

In this appendix we compare the performance of the ML estimators when initialised with the Silverman rule-of-thumb and MLE rule-of-thumb estimators. We restrict our comparison to CV and RW class-specific PCA1 data, since the purpose of this appendix is not to perform a comprehensive study of bandwidth initialisation procedures, but rather to highlight the implications of selecting different bandwidth initialisation procedures and to confirm that the regularities in the relative performance of the ML estimators when initialised with the MLE rule-of-thumb estimator are consistent with the results of the ML estimators initialised with the Silverman rule-of-thumb estimator, as presented in Chapter 6.

B.2 COMPARISON OF MLE RULE-OF-THUMB AND SILVERMAN RULE-OF-THUMB INITIALISATION

B.2.1 CV DATASET RESULTS

Table B.1 shows the comparative CV class-specific PCA1 entropy results of the ML estimators initialised with the MLE rule-of-thumb estimator. The dataset name is indicated with "DS", the class number with "C", the dimensionality of the class after pre-processing with "K", and we abbreviate the dataset names with two-letter acronyms. We make use of color scales to visually represent the relative performance of the estimators, where green indicates the lowest entropy for a specific class and red the highest entropy for a specific class. We use the same experimental design as in Section 6.3.2, except for the initialisation procedure.

If we compare the results of the ML estimators initialised with the MLE rule-of-thumb bandwidths in Table B.1 to the results of the ML estimator initialised with the Silverman rule-of-thumb bandwidths in Table 6.1, we observe that the results are similar for the MLE(g), ML-LOO(f) and ML-LOO(s) estimators. However, the MLE and MLL estimators experience performance improvements for most classes and for some classes drastic improvements are observed, for example, on classes 1 and 3 of the Balance Scale dataset.

From the definition of entropy, we know that the entropy score is similar to the product of the likelihood scores, thus a single likelihood score can have a significant effect on the entropy score. We therefore investigate the entropy scores of the Balance Scale dataset classes 1 and 3, by removing the effect on the entropy score one sample at

Table B.1: CV entropy results (MLE initialisation)(class-specific PCA1)

DS	C	K	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
OF	1	2	1.4791	1.4597	1.6042	1.4722	1.4856
BS	1	4	5.4498	-9.6157	-4.5769	5.4532	5.4680
	2	3	4.8351	4.9986	5.0003	4.8615	4.8717
	3	4	5.4336	-10.4157	-5.0491	5.4287	5.4636
IR	1	4	5.5401	5.4680	5.4661	5.5738	5.6912
	2	4	4.7940	4.8085	4.8085	4.8136	4.9217
	3	4	5.0203	5.0253	5.0323	5.0464	5.2165
DB	1	8	9.8402	9.7251	9.7455	9.6812	9.9478
	2	8	10.4846	10.3122	10.3793	10.3283	10.4337
HS	1	13	18.1078	18.3467	18.3467	15.8792	18.3481
	2	13	17.5534	17.5955	17.6582	14.9996	17.7300
VC	1	9	10.5107	10.8497	11.0679	10.3810	11.2042
	2	8	9.7975	10.0659	10.2778	9.6309	10.4234
	3	8	8.9019	8.7250	8.7453	8.4708	9.2330
	4	11	11.1391	11.1859	11.4430	11.7201	12.0935
WF	1	21	26.7618	26.5689	26.5689	26.7907	27.4237
	2	21	29.2118	29.0202	29.0202	29.2533	29.5631
	3	21	29.1504	28.9282	28.9282	29.1900	29.5093
IS	1	33	42.3937	41.4014	42.2876	42.3950	45.7906
	2	12	13.5817	12.2670	12.7409	13.6711	14.2113

a time. We first sort the likelihood scores of all test data points (from the 10 test folds) in a class from highest to lowest and then systematically remove the highest likelihood score (without replacement). We re-calculate the entropy score after each removal and plot the entropy. Figures B.1 and B.2 illustrate the entropy graphs obtained in this manner, for classes 1 and 3 of the Balance Scale dataset.

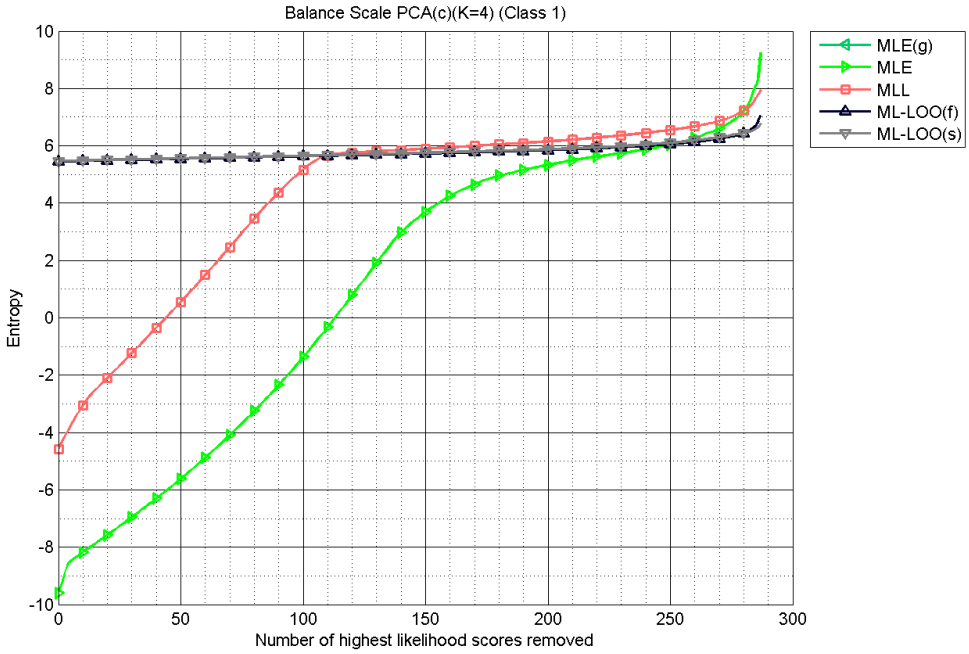

Figure B.1: *Balance Scale entropy results as likelihoods are removed (Class 1, class-specific PCA1)*

We observe very similar behaviour in Figures B.1 and B.2 for all estimators. The MLE performs optimally, followed by the MLL estimator. For the MLE estimator, at least 250 of the highest likelihood scores must be removed to yield an entropy score similar to the MLE(g), ML-LOO(s) and ML-LOO(f) estimators, and for the MLL estimator at least 100 of the highest likelihood scores must be removed to yield an entropy score similar to the MLE(g), ML-LOO(s) and ML-LOO(f) estimators. These results show that the superior entropy score of the MLE and MLL estimators are the result of a significant number of test data points with superior likelihood scores, and not the result of only a few test data points with exceptionally large likelihood scores. We note that this superior performance may be caused by over fitting, since no independent test sets were provided for the CV datasets, and the optimal model was selected and tested on the same cross-validation test folds. There is thus not an independent test set to verify the generalisation performance of the optimal model selected and if data points

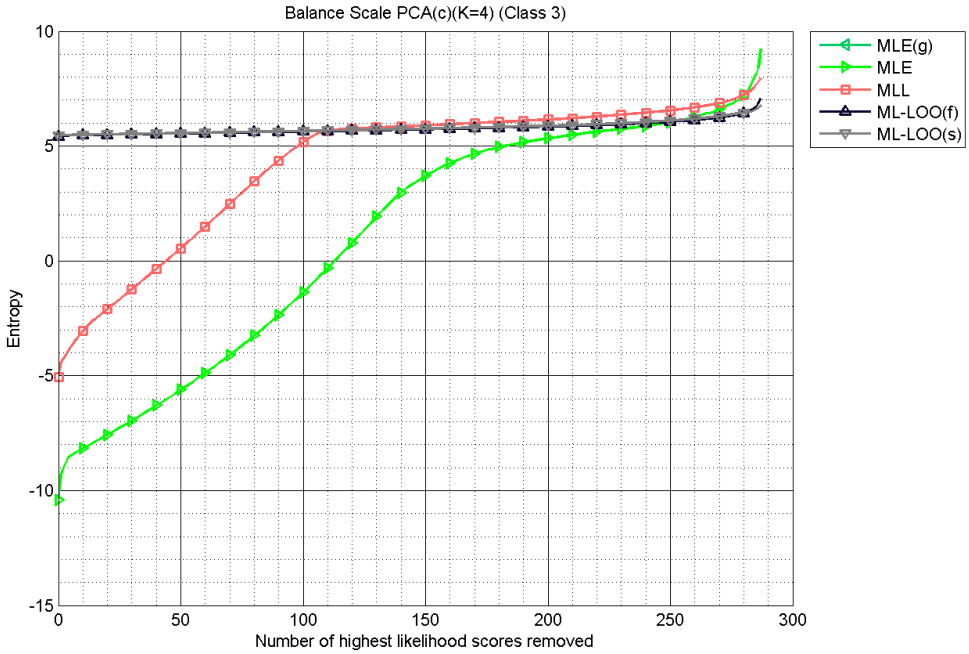

Figure B.2: *Balance Scale entropy results as likelihoods are removed (Class 3, class-specific PCA1)*

between different folds of the dataset are very close in feature space, the likelihood scores of data points close to each other can be significantly increased be decreasing the bandwidths of these data points.

Since the MLE and MLL estimators estimate a unique bandwidth per kernel, as opposed to the other estimators that estimate an identical kernel bandwidth for all kernels,they have more free parameters and are more susceptible to overtraining. In the next section we will verify whether this superior performance is indeed the result of overtraining by testing whether the MLE and MLL estimators show similar superior performance on the RW datasets with independent test sets. If similar superior performance of the MLE and MLL estimators is not observed on the RW datasets (with independent training and test sets), then this indicates that the MLE and MLL estimators are susceptible to overtraining when CV is used, and that the MLE rule-of-thumb initialisation increases the effect of overtraining, since the same degree of overtraining is not observed for the CV results in Table 6.2. For example, we observe in Table 6.2 that the Balance Scale entropy scores of the MLE estimator on classes 1 and 3 are -1.0283 and -1.2970 when initialised with the Silverman rule-of-thumb bandwidths, whereas the entropy results in Table B.1 for the same classes are -4.579 and -5.0491 respectively when the MLE bandwidths are initialised with the MLE rule-of-thumb

bandwidths.

B.2.2 RW DATASET RESULTS

Table B.2 shows the comparative class-specific PCA1 entropy results of the ML estimators initialised with the MLE rule-of-thumb estimator for the Letter dataset.

Table B.2: Letter test entropy results (MLE initialisation)(class-specific PCA1)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	15	789	9.2839	10.1071	10.3585	8.0562	10.1490
2	15	766	11.8519	12.3283	12.4573	10.6145	12.5777
3	15	736	10.9051	11.2052	11.3901	10.2583	11.8198
4	14	805	10.6827	11.1028	11.1752	9.9238	11.1285
5	15	768	9.0923	9.9453	10.0564	7.4655	9.8397
6	13	775	9.3750	9.6458	9.7526	8.3397	10.0095
7	15	773	11.6828	12.0998	12.2476	10.6827	12.3795
8	14	734	10.6841	10.5431	10.5522	8.9877	11.4681
9	13	755	8.7685	6.6297	6.1548	7.3886	9.1985
10	13	747	7.9987	8.1356	8.2729	6.9327	8.5561
11	15	739	10.1304	10.5326	10.6065	9.1376	11.0120
12	12	761	6.9569	6.9068	6.8784	5.9943	7.4229
13	14	792	8.3513	9.1453	9.2862	6.8357	9.0251
14	14	783	9.3556	9.5293	8.2142	8.4246	10.1888
15	16	753	12.7934	12.8139	12.6994	11.5766	13.7354
16	15	803	9.9916	10.9968	11.1859	9.4706	10.8269
17	15	783	10.8592	11.5855	11.6745	9.7270	11.8547
18	15	758	10.9090	11.6637	11.7861	10.0001	11.3790
19	15	748	9.0289	9.9074	9.8645	7.5860	9.7707
20	13	796	7.9007	8.2578	8.4002	6.7434	8.7922
21	13	813	8.8813	8.9520	8.9088	7.9459	9.2364
22	14	764	10.0594	10.3158	10.6220	8.7746	10.8550
23	15	752	10.5563	10.8136	11.0401	9.3770	11.8689
24	15	787	11.0388	10.9528	10.8425	9.7799	12.2916
25	14	786	8.6997	9.1915	9.3148	7.4566	9.2300
26	15	734	10.2944	9.8884	9.9282	9.1276	11.2879

If we compare the results of the ML estimators initialised with the MLE rule-of-thumb bandwidths in Table B.2 to the results of the ML estimators initialised with the Silverman rule-of-thumb bandwidths in Table 6.4, we observe that the results are similar for the MLE(g), ML-LOO(f) and ML-LOO(s) estimators. This is consistent

with our observations on the CV results in Table B.1. An important observation is that although the MLE and MLL estimator experience relative improvements in performance, the ML-LOO(f) estimator still performs optimally on most classes, and the relative performances between all estimators remain consistent – irrespective of the initialisation procedure.

Table B.3 shows the comparative class-specific PCA1 entropy results of the ML estimators initialised with the MLE rule-of-thumb estimator for the Segmentation dataset.

Table B.3: Segmentation test entropy results (MLE initialisation)(class-specific PCA1)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	13	330	14.7862	14.4223	14.1016	14.0612	16.6909
2	11	330	13.4029	11.9142	11.6911	8.1211	13.8239
3	12	330	10.8952	2.2743	1.9325	8.5740	9.7009
4	11	330	15.3963	7.7054	7.2726	4.8036	13.2683
5	11	330	15.5017	7.7815	6.5676	9.2814	17.6808
6	11	330	11.9213	10.6857	10.6653	10.8369	12.2270
7	10	330	7.1897	6.2268	6.5207	3.5381	7.4221

We observe that the MLL estimator in Table B.3 experiences relative performance improvements on all classes and interestingly the ML-LOO(f) estimator experiences relative performance improvements on six of the seven classes, compared to the results in Table 6.8. The MLE estimator, however, only experiences a performance improvement on class 7. The MLE estimator thus experienced degradation in performance for the MLE rule-of-thumb initialisation since the MLE performed optimally on six classes in Table 6.8 and performs optimally on only two classes in Table B.3.

Table B.4 shows the comparative class-specific PCA1 entropy results of the ML estimators initialised with the MLE rule-of-thumb estimator for the Landsat dataset.

Table B.4: Landsat test entropy results (MLE initialisation)(class-specific PCA1)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	8	1533	10.6458	10.5521	10.8456	10.5672	11.0781
2	10	703	11.4595	11.0589	11.2062	11.3104	12.0685
3	14	1358	16.6453	16.3367	16.4281	16.4253	17.8978
4	13	626	14.6365	14.5462	14.4505	14.7884	15.7918
5	11	707	14.1894	14.2939	14.1562	14.1767	14.8323
6	11	1508	14.3085	13.8502	14.1883	14.2131	15.1071

We observe that the MLL estimator in Table B.4 experiences relative performance improvements on six of the seven classes, and the MLE estimator on all classes compared to the results in Table 6.12. The relative performances of the remaining estimators remain consistent. We also observe that the MLE estimator experienced a relative increase in performance compared to the ML-LOO(f) estimator, since the MLE performed optimally on only two classes for the Silverman initialisation and performs optimally on four classes for the MLE initialisation. Conversely, the ML-LOO(f) estimator performed optimally on five of the six classes for Silverman initialisation and performs optimally on only one class for the MLE initialisation.

Table B.5 shows the comparative class-specific PCA1 entropy results of the ML estimators initialised with the MLE rule-of-thumb estimator for the Optdigits dataset.

Table B.5: Optdigits test entropy results (MLE initialisation)(class-specific PCA1)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	46	554	52.6667	53.8426	53.8426	53.6575	57.2346
2	36	571	37.9291	40.1827	40.1827	36.8352	39.8576
3	41	557	55.1568	55.7373	55.7373	57.5336	55.8896
4	50	572	63.7816	63.4517	63.4517	76.7163	69.8444
5	39	568	43.0227	45.2993	45.2993	37.2329	44.7705
6	46	558	55.7480	57.0661	57.0661	57.2695	59.6361
7	37	558	46.8943	46.9708	46.9708	44.3374	48.7581
8	41	566	52.9535	53.4834	53.4834	50.9872	56.0250
9	50	554	61.2927	62.8508	62.8508	62.0969	65.3551
10	43	562	58.1301	57.4619	57.4619	50.8143	59.1577

We observe that the MLE and MLL estimators in Table B.5 experience relative performance improvements on three of the 10 classes, the ML-LOO(f) estimator in seven of the 10 classes and the MLE(g) on only one class, compared to the results in Table 6.16. The relative performances of the remaining estimators remain consistent. We also observe that the ML-LOO(f) estimator experienced a relative improvement in performance, since this estimator performed optimally on only one class for the Silverman initialisation and performed optimally on five classes for the MLE initialisation.

Table B.6 shows the comparative class-specific PCA1 entropy results of the ML estimators initialised with the MLE rule-of-thumb estimator for the Isolet dataset.

We observe that the MLE and MLL estimators in Table B.6 experience relative performance improvements on most classes, compared to the results in Table 6.20 and the relative performances of the remaining estimators remain mostly consistent. For

Table B.6: Isolet test entropy results (MLE initialisation)(class-specific PCA1)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	92	300	189.0757	187.3541	187.3541	189.1424	205.5436
2	107	300	211.6831	206.4788	206.4788	211.0833	227.0845
3	98	300	203.3344	197.1934	197.1934	200.8515	218.5132
4	101	300	205.8297	200.7884	200.7884	205.3712	220.4028
5	103	300	213.4577	214.7763	214.7763	215.2791	230.2526
6	100	300	199.3415	197.4903	197.4903	199.3451	216.9816
7	90	300	186.8601	183.4525	183.4525	186.5587	198.8575
8	88	300	180.7644	177.7234	177.7234	180.7331	195.6963
9	108	300	230.3365	232.5091	232.5091	233.5874	242.9945
10	85	300	182.6356	180.4786	180.4786	182.2690	193.5533
11	89	300	188.6171	185.8624	185.8624	188.3154	199.9651
12	118	300	241.4118	247.0241	247.0241	245.1218	260.2248
13	123	300	244.2866	244.0540	244.0540	246.5306	265.4168
14	119	300	235.5346	235.7297	235.7297	238.8468	258.6651
15	103	300	209.8113	209.4096	209.4096	210.3239	228.0077
16	104	300	206.8461	200.9151	200.9151	206.0977	221.1179
17	108	300	222.7288	226.0176	226.0176	224.0843	240.0169
18	106	300	217.0423	218.1320	218.1320	218.1609	235.8836
19	96	300	199.3797	199.3623	199.3623	200.3635	215.3954
20	100	300	202.9346	196.5860	196.5860	201.4843	217.6514
21	121	300	239.3337	235.8053	235.8053	240.6877	256.6692
22	113	300	226.9840	223.8469	223.8469	227.1875	248.2845
23	104	300	207.1395	203.8252	203.8252	207.2196	220.9296
24	92	300	194.4145	196.2914	196.2914	195.1491	209.3186
25	109	300	218.3484	213.5680	213.5680	218.2181	235.8036
26	100	300	206.8304	201.6605	201.6605	204.8902	224.3891

all classes where the MLE and MLL estimators perform optimally they have identical entropy scores - this indicates that these entropy scores were obtained at initialisation and that the superior performance of these estimators are actually caused by the superior performance of the MLE rule-of-thumb estimator. We also observe that the MLE(g) estimator performed optimally on 21 of the 26 classes for the Silverman initialisation, and performs optimally on only 7 of the 26 classes for the MLE initialisation. Interestingly, the results for the MLE and MLL estimators are exactly the same for all classes when initialised with the MLE, we confirmed that the optimal entropy results for these estimators were at initialisation and that for both estimators the CV entropy on the training set and the test set entropy increased after initialisation similar to, for example, the MLL results for class 3 of the Landsat dataset in Figure 6.20. The seemingly superior performance of the MLE and MLL estimators is thus because of the good MLE rule-of-thumb initialisation for these estimators, and the estimators do not improve the performance with additional iterations.

B.3 CONCLUSION

In Section B.1 we observed on the CV datasets that the MLE and MLL estimators experienced relative improvements in performance on most classes when using the MLE rule-of-thumb bandwidths for initialisation as opposed to the Silverman rule-of-thumb bandwidths. The ML-LOO(f) estimator experienced relative performance improvements on some classes and a degradation in performance on other classes, while the MLE(g) and ML-LOO(s) estimators showed relatively small performance variations between the two initialisation methods. We investigated the significant difference in performance of the MLE and MLL estimators compared to the other estimators on classes 1 and 3 of the Balance Scale dataset, and motivated that the large margin in superior performance is probably caused by the fact that the results were obtained with CV and not with an independent test set as in the case of the RW datasets.

We showed in Section B.2 that the MLE and MLL estimator did not outperform the other estimators on any class with such a large margin as in the case of classes 1 and 3 of the CV Balance Scale dataset. We therefore suspect that the superior performance of the MLE and MLL estimators on some CV dataset might be a consequence of overtraining and might thus not generalise to truly unrelated test cases.

In Section B.2 we observed on the RW datasets that the MLE and MLL estimators again experienced relative improvements in performance on most classes when using the MLE rule-of-thumb bandwidths for initialisation as opposed to the Silverman rule-

of-thumb bandwidths. We also observed that the ML-LOO(f) estimator experiences relative improvements in performance on the Segmentation and Optdigits datasets, and a performance degradation on the Letter dataset. On the Letter dataset, the ML-LOO(f) estimator performed optimally on most classes for both initialisations; on the Segmentation dataset the MLE estimator performed optimally on most classes for Silverman initialisation and the ML-LOO(f) estimator performed optimally on most classes for the MLE initialisation. On the Landsat dataset, the ML-LOO(f) estimator performed optimally on most classes for the Silverman initialisation and the MLE estimator performed optimally on most classes for the MLE initialisation. On the Optdigits dataset the MLE(g) estimator performed optimally on most classes for the Silverman initialisation and performed well (together with the ML-LOO(f) estimator) on most classes for the MLE initialisation. On the Isolet dataset the MLE(g) performed optimally on most classes for the Silverman initialisation and the MLE and MLL algorithms performed optimally on most classes for the MLE initialisation.

These results show that the MLE rule-of-thumb estimator does in fact frequently yield better performance for the MLE and MLL estimators, and that this improved performance does not hold for the ML-LOO(s) and MLE(g) estimators, and holds only in some cases for the ML-LOO(f) estimator.

However, since Silverman initialisation gives more consistent performance behaviour across the different ML estimators compared, and does not give a clear advantage to the MLL and MLE estimators, we prefer to use the Silverman estimator as the bandwidth initialisation algorithm for our comparative experiments. We do, however, recognise that bandwidth initialisation has a notable effect on the results of ML estimators, and that other estimators might be even more advantageous than the Silverman and MLE rule-of-thumb estimators considered. It is also possible that no bandwidth initialisation procedure will be optimal for all types of data; therefore this will require a comprehensive study of its own, and is a matter for further research. The focus of this study is on gaining better understanding of the relative performances of the ML estimators considered, and therefore we prefer the consistency of the Silverman estimator for bandwidth initialisation.

Appendix C

Maximum-likelihood Estimator Results (Global PCA)

Contents

C.1 Introduction 192

C.2 CV Global PCA Results 193

C.3 RW Global PCA Results 195

C.4 Convergence Examples of ML Estimators 207

C.5 Conclusion 212

C.1 INTRODUCTION

In Chapter 6 we presented the comparative entropy results of the MLE(g), MLE, MLL, ML-LOO(f) and ML-LOO(s) estimators for class-specific PCA5 and PCA1 datasets. In this appendix, we present similar comparative results for data pre-processed with the global PCA transformation (i.e. where the data of all classes are pre-processed simultaneously and not separately). We also compare the CV and test performance of the ML estimators over the number of training iterations to illustrate the convergence of the estimators on different tasks.

The results for the global PCA5 and PCA1 CV datasets are presented in Section C.2; the results for the global PCA5 and PCA1 RW datasets are presented in Section

C.3 and the entropy results of the ML estimators over training iterations are presented in Section C.4. We make concluding remarks regarding the difference in estimator performance for class-specific and globally pre-processed data in Section C.5.

C.2 CV GLOBAL PCA RESULTS

Table C.1: CV entropy results (global PCA5)

DS	C	K	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
OF	1	2	1.4815	1.4641	1.4653	1.4781	1.4946
BS	1	4	5.0717	3.8911	4.3158	5.0446	5.0842
	2	4	5.3998	5.4046	5.4046	4.2193	5.3923
	3	4	5.0303	3.9025	4.2922	5.0243	5.0584
IR	1	4	0.3640	0.4805	0.0809	0.3311	1.2622
	2	4	1.3247	1.3247	1.3247	1.3272	1.7807
	3	4	2.3421	2.3421	2.3421	2.3077	2.6583
DB	1	5	6.6377	6.5715	6.6001	6.6500	6.6841
	2	5	7.6831	7.7030	7.7047	7.6794	7.6768
HS	1	5	7.8999	7.9098	7.9366	7.8393	7.8789
	2	5	7.7205	7.6662	7.7293	7.5638	7.6836
VC	1	5	7.5296	7.5152	7.5361	7.5238	7.7480
	2	5	7.4860	7.4502	7.4780	7.4614	7.6873
	3	5	6.5442	6.7565	6.4202	6.6851	6.4655
	4	5	6.9685	6.8094	6.9685	6.9412	6.9734
WF	1	5	7.6989	7.7245	7.7245	7.6992	7.7094
	2	5	7.4130	7.4708	7.4585	7.3917	7.4023
	3	5	7.4380	7.4781	7.4714	7.4131	7.4265
IS	1	5	10.0059	10.0409	10.2469	10.0633	10.0376
	2	5	6.1999	5.9269	5.8518	6.1799	6.2873

Table C.2: CV entropy results (global PCA1)

DS	C	K	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
OF	1	2	1.4815	1.4641	1.4653	1.4781	1.4946
BS	1	4	5.0717	3.8911	4.3158	5.0446	5.0842
	2	4	5.3998	5.4046	5.4046	4.2193	5.3923
	3	4	5.0303	3.9025	4.2922	5.0243	5.0584
IR	1	3	1.1139	1.1175	1.1870	1.1083	1.6499
	2	3	2.0303	2.0363	2.0363	2.0310	2.1371
	3	3	2.5697	2.5697	2.5697	2.5585	2.6456
DB	1	8	9.0959	9.0731	9.1325	8.9352	9.1808
	2	8	10.8040	10.8747	10.9021	10.7716	10.7945
HS	1	13	18.0056	18.2799	18.4083	14.0344	18.1098
	2	13	16.2138	16.4577	16.4577	15.1323	16.3107
VC	1	9	9.3899	9.6916	9.7765	8.9755	10.1689
	2	9	9.1220	9.3322	9.4815	8.3568	10.0788
	3	9	8.1371	8.1282	7.9935	7.6488	9.4209
	4	9	9.0398	8.7603	9.0526	8.9208	9.1690
WF	1	21	24.9552	25.3210	25.3210	24.9612	25.5374
	2	21	24.8054	25.2675	25.2675	24.7123	25.1262
	3	21	24.9187	25.4349	25.4349	24.8139	25.2488
IS	1	32	49.3211	48.9020	53.1074	53.1453	51.7992
	2	32	3.8457	-0.5367	7.2051	3.4136	6.9623

C.3 RW GLOBAL PCA RESULTS

Table C.3: Letter test entropy results (global PCA5)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	5	789	4.0880	4.0618	4.0900	3.9857	4.2397
2	5	766	4.4272	4.4858	4.4812	4.3572	4.9370
3	5	736	4.7731	4.6082	4.7371	4.7302	4.8610
4	5	805	4.3461	4.3267	4.3845	4.2842	4.5026
5	5	768	4.2930	4.3701	4.4288	4.1231	4.5013
6	5	775	4.9626	4.8318	4.8370	4.8033	5.0838
7	5	773	4.7121	4.6926	4.6686	4.6421	4.8458
8	5	734	4.4389	4.4703	4.4655	4.3260	4.7025
9	5	755	3.5859	3.2418	3.2154	3.4715	4.9006
10	5	747	4.9097	4.9078	5.0175	4.6910	5.0588
11	5	739	4.8489	4.7704	4.7643	4.7753	5.1093
12	5	761	4.5094	4.3252	4.3699	4.3749	4.6698
13	5	792	4.9695	5.0054	5.0799	4.8081	5.3559
14	5	783	4.4686	4.3263	4.3244	4.3530	4.9314
15	5	753	4.1657	4.1559	4.1666	4.1602	4.3431
16	5	803	5.1434	5.2004	5.2020	5.1148	5.3473
17	5	783	4.9647	5.0337	5.0683	4.9164	5.1855
18	5	758	4.2484	4.2967	4.3169	4.2069	4.3891
19	5	748	4.3807	4.4318	4.4149	4.2794	4.6714
20	5	796	4.5710	4.5823	4.6261	4.4731	4.8116
21	5	813	4.2179	4.1446	4.2217	4.1697	4.4335
22	5	764	3.8193	3.7591	3.8067	3.7281	4.0594
23	5	752	5.0081	5.1391	5.1450	4.9444	5.4080
24	5	787	4.3654	4.3706	4.3036	4.2201	4.7040
25	5	786	4.9398	4.9164	4.9715	4.8539	5.1509
26	5	734	4.6133	4.3580	4.4561	4.5351	4.9161

Table C.4: Letter test entropy results (global PCA1)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	16	789	4.9680	5.6431	5.6744	3.4714	6.0986
2	16	766	5.4892	5.7428	6.1824	3.9922	6.2779
3	16	736	5.5469	5.5330	5.5343	4.3942	6.4143
4	16	805	4.6576	4.8866	5.0856	3.6990	5.5053
5	16	768	4.3192	4.5326	4.6792	2.8250	5.2816
6	16	775	5.1597	5.2613	5.1569	4.0331	6.0690
7	16	773	6.2129	6.8173	6.9593	5.0988	6.6914
8	16	734	5.8387	5.9901	6.0634	4.0197	7.0463
9	16	755	2.5944	1.8990	1.9194	1.1661	3.4184
10	16	747	5.0055	5.1695	5.1423	3.6313	5.6700
11	16	739	6.7691	6.9698	7.1124	5.7621	7.7322
12	16	761	4.7266	4.8214	4.7733	3.6339	5.5692
13	16	792	6.1479	6.1078	6.3934	4.5447	7.5130
14	16	783	5.4640	5.0749	4.8263	4.3939	6.5482
15	16	753	4.5659	4.9169	5.2170	3.5174	5.1242
16	16	803	6.0274	6.5295	6.7330	5.2372	7.0169
17	16	783	6.6242	7.0892	7.1931	5.3494	8.0633
18	16	758	5.3725	5.8630	5.7499	4.4403	6.1516
19	16	748	5.0435	5.3111	5.7553	3.5336	5.7707
20	16	796	4.4441	4.9228	4.9503	3.2380	5.4605
21	16	813	5.0613	5.2453	5.3264	4.0059	5.8970
22	16	764	4.7432	5.1204	5.3265	3.2760	5.6601
23	16	752	6.1765	6.5310	6.7846	3.9968	8.0031
24	16	787	4.8942	4.8481	4.9278	3.2593	6.2510
25	16	786	6.0730	6.5970	6.4980	4.8763	7.1285
26	16	734	5.0860	5.4515	5.7159	3.8378	6.1453

Table C.5: Letter statistical difference test results (global PCA5)

C	K	NC	TE	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	5	789	3	0	0	0	-	0
2	5	766	3	0	0	0	-	1
3	5	736	3	0	-	0	0	0
4	5	805	3	0	0	0	-	0
5	5	768	3	0	0	0	-	0
6	5	775	3	0	0	0	-	0
7	5	773	3	0	0	0	-	0
8	5	734	3	0	0	0	-	0
9	5	755	1	1	0	-	1	1
10	5	747	3	0	0	0	-	1
11	5	739	3	0	0	-	0	0
12	5	761	3	0	-	0	0	0
13	5	792	3	0	0	0	-	1
14	5	783	3	0	0	-	0	0
15	5	753	3	-	0	0	0	0
16	5	803	3	0	0	0	-	0
17	5	783	3	0	0	0	-	0
18	5	758	3	0	0	0	-	0
19	5	748	3	0	0	0	-	0
20	5	796	3	0	0	0	-	0
21	5	813	3	0	-	0	0	0
22	5	764	3	0	0	0	-	1
23	5	752	3	0	0	0	-	1
24	5	787	3	0	0	0	-	1
25	5	786	3	0	0	0	-	0
26	5	734	3	0	-	0	0	0

Table C.6: Letter statistical difference test results (global PCA1)

C	K	NC	TE	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	16	789	3	1	0	1	-	0
2	16	766	1	1	1	1	-	1
3	16	736	3	1	1	1	-	0
4	16	805	3	1	1	0	-	1
5	16	768	3	1	0	0	-	0
6	16	775	2	0	0	0	-	1
7	16	773	3	1	1	1	-	1
8	16	734	3	1	0	0	-	0
9	16	755	3	1	0	0	-	0
10	16	747	3	1	1	0	-	1
11	16	739	3	1	0	0	-	0
12	16	761	3	0	1	0	-	0
13	16	792	3	1	0	0	-	1
14	16	783	3	1	0	0	-	0
15	16	753	3	0	1	0	-	0
16	16	803	3	0	0	1	-	0
17	16	783	3	1	1	1	-	1
18	16	758	3	1	0	0	-	0
19	16	748	1	1	1	1	-	1
20	16	796	3	1	1	1	-	1
21	16	813	3	1	1	1	-	1
22	16	764	3	1	1	0	-	1
23	16	752	3	1	1	1	-	1
24	16	787	3	1	0	0	-	0
25	16	786	3	1	1	1	-	1
26	16	734	3	0	0	0	-	1

Table C.7: Segmentation test entropy results (global PCA5)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	5	330	2.3017	2.2266	2.3706	1.9706	2.2902
2	5	330	3.4112	3.4995	2.9079	2.5082	4.2487
3	5	330	2.5371	0.1336	0.4054	2.4492	2.3935
4	5	330	3.8852	3.1247	3.2680	3.8043	3.9319
5	5	330	7.6962	3.6320	3.9032	7.6590	8.2734
6	5	330	-0.0955	0.0527	0.1962	0.8145	-0.1446
7	5	330	-0.0780	-0.9630	-0.7859	0.1651	-0.1734

Table C.8: Segmentation test entropy results (global PCA1)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	13	330	1.1732	-0.3325	-0.1548	1.1732	1.3198
2	13	330	0.6258	3.6090	5.2923	1.6428	4.5516
3	13	330	0.2193	-6.2477	-4.5293	4.3785	1.5777
4	13	330	12.8055	9.0327	12.9528	13.5635	10.2183
5	13	330	21.9523	21.9523	5.9553	33.9773	31.0458
6	13	330	-2.6906	-4.4826	-4.6384	0.0030	-2.5681
7	13	330	-8.7849	-8.5209	-8.7112	-23.8403	-7.6394

Table C.9: Segmentation statistical difference test results (global PCA5)

C	K	NC	TE	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	5	330	3	0	0	0	-	0
2	5	330	3	0	0	0	-	0
3	5	330	3	1	-	0	1	0
4	5	330	3	0	-	0	0	0
5	5	330	3	1	-	0	0	1
6	5	330	3	-	0	0	0	0
7	5	330	1	1	-	0	1	1

Table C.10: Segmentation statistical difference test results (global PCA1)

C	K	NC	TE	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	13	330	1	1	-	0	1	1
2	13	330	3	-	0	0	1	0
3	13	330	1	1	-	0	1	1
4	13	330	3	1	-	0	1	1
5	13	330	3	1	1	-	1	1
6	13	330	1	1	0	-	1	1
7	13	330	1	1	1	1	-	1

Table C.11: Landsat test entropy results (global PCA5)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	5	1533	4.8139	4.8046	4.8286	4.7444	5.0022
2	5	703	5.9260	5.7680	5.8026	5.7423	6.3208
3	5	1358	3.6676	3.6976	3.7447	3.5325	3.9484
4	5	626	4.1328	3.9694	4.0405	3.9114	4.3172
5	5	707	6.4714	6.5138	6.4623	6.4902	6.8060
6	5	1508	4.1157	4.0732	4.1171	4.0563	4.3587

Table C.12: Landsat test entropy results (global PCA1)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	8	1533	6.1385	6.2419	6.3429	5.9897	6.6426
2	8	703	8.0974	7.9978	7.9198	7.7103	8.8443
3	8	1358	4.8316	4.7894	4.7554	4.5217	5.4330
4	8	626	5.1371	5.2007	5.3579	4.9495	5.6109
5	8	707	8.6172	8.7447	8.7798	8.6181	9.1252
6	8	1508	5.2880	5.1039	5.3391	5.1248	5.8552

Table C.13: Landsat statistical difference test results (global PCA5)

C	K	NC	TE	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	5	1533	3	0	0	0	-	1
2	5	703	3	0	0	0	-	1
3	5	1358	3	0	0	0	-	1
4	5	626	3	0	0	0	-	1
5	5	707	3	0	0	-	0	0
6	5	1508	3	0	0	0	-	1

Table C.14: Landsat statistical difference test results (global PCA1)

C	K	NC	TE	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	8	1533	3	0	1	1	-	1
2	8	703	3	1	0	0	-	1
3	8	1358	3	1	0	0	-	1
4	8	626	3	0	0	0	-	1
5	8	707	3	0	0	0	-	0
6	8	1508	3	0	-	0	0	1

Table C.15: Optdigits test entropy results (global PCA5)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	5	554	5.9358	5.9733	6.0502	5.9228	5.9482
2	5	571	7.4323	7.4416	7.4754	7.4508	7.4056
3	5	557	8.0508	7.9038	7.8826	7.9590	8.2283
4	5	572	7.6085	7.5700	7.6496	7.5221	7.6541
5	5	568	8.4995	8.4617	8.4336	8.3718	8.4651
6	5	558	8.0324	8.0347	8.0347	8.0285	7.9960
7	5	558	6.8092	6.7119	6.7533	6.6912	6.8557
8	5	566	7.5076	7.4534	7.4758	7.5347	7.5941
9	5	554	7.8588	7.8588	7.8588	7.8447	7.8454
10	5	562	7.6240	7.6583	7.6133	7.6134	7.5816

Table C.16: Optdigits test entropy results (global PCA1)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	60	554	30.3421	32.2522	32.2522	32.2529	33.8393
2	60	571	41.6246	41.6246	41.6246	41.5517	43.2299
3	60	557	53.9065	53.9065	53.9065	53.4374	57.1524
4	60	572	50.1513	50.1870	50.1870	50.8373	53.6511
5	60	568	58.4932	58.4932	58.4932	59.3826	60.2437
6	60	558	52.4742	52.4742	52.4742	52.4564	53.9241
7	60	558	34.8564	38.6712	38.6712	38.9048	38.9006
8	60	566	47.8951	49.9260	49.9260	49.9243	51.8968
9	60	554	50.3352	50.3352	50.3352	54.6098	53.5366
10	60	562	50.1016	51.4106	51.4106	51.4175	53.9883

Table C.17: Optdigits statistical difference test results (global PCA5)

C	K	NC	TE	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	5	554	3	0	0	0	-	0
2	5	571	3	0	0	0	0	-
3	5	557	3	0	0	-	0	0
4	5	572	3	0	0	0	-	0
5	5	568	3	0	0	0	-	0
6	5	558	3	0	0	0	0	-
7	5	558	3	0	0	0	-	0
8	5	566	3	0	-	0	0	0
9	5	554	3	0	0	0	-	0
10	5	562	3	0	0	0	0	-

Table C.18: Optdigits statistical difference test results (global PCA1)

C	K	NC	TE	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	60	554	3	-	1	1	1	1
2	60	571	3	0	0	0	-	1
3	60	557	3	0	0	0	-	1
4	60	572	3	-	1	1	1	1
5	60	568	3	-	0	0	0	1
6	60	558	3	0	0	0	-	0
7	60	558	3	-	1	1	1	1
8	60	566	3	-	1	1	1	1
9	60	554	3	-	0	0	0	0
10	60	562	3	-	1	1	1	1

Table C.19: Isolet test entropy results (global PCA5)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	5	300	13.0863	13.0951	13.1545	13.0723	13.1017
2	5	300	12.8671	12.9635	12.9453	12.8105	12.8780
3	5	300	13.0416	13.0650	13.0650	13.0475	13.0389
4	5	300	13.0601	13.0173	13.0190	13.2104	13.1160
5	5	300	13.3311	13.3203	13.3014	13.3864	13.3605
6	5	300	12.5353	12.5353	12.5353	12.5353	12.5958
7	5	300	12.6008	12.6389	12.6299	12.5868	12.6254
8	5	300	12.8328	12.8457	12.8943	12.8308	12.8494
9	5	300	12.8927	12.9389	12.9389	12.8674	12.9400
10	5	300	12.5076	12.5604	12.4981	12.4904	12.5449
11	5	300	12.2917	12.4003	12.3633	12.2254	12.2909
12	5	300	12.8060	12.7974	12.7974	12.8621	12.8609
13	5	300	12.9958	13.0223	13.0223	12.9765	13.0454
14	5	300	12.9126	12.8362	12.8362	12.9029	12.9385
15	5	300	13.4747	13.4837	13.4837	13.4272	13.4481
16	5	300	12.6930	12.7796	12.7261	12.6836	12.7151
17	5	300	12.8691	12.8910	12.8910	12.8580	12.8764
18	5	300	12.7414	12.7024	12.7024	12.7361	12.8249
19	5	300	12.6569	12.5997	12.5997	12.6681	12.6284
20	5	300	13.0398	13.2217	13.1329	13.0405	13.0803
21	5	300	13.0067	13.0067	13.0067	13.0237	13.0623
22	5	300	13.7971	13.7915	13.7915	13.7591	13.7467
23	5	300	14.5195	14.5122	14.5122	14.5263	14.5262
24	5	300	12.4988	12.5197	12.5641	12.4780	12.4966
25	5	300	12.1015	12.1507	12.1507	11.9941	12.0949
26	5	300	13.8779	13.8779	13.8779	13.9428	13.9093

Table C.20: Isolet test entropy results (global PCA1)

C	K	NC	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	64	300	124.0725	124.7566	124.7566	119.7868	128.3926
2	64	300	123.1627	123.7344	123.7344	123.7344	127.1733
3	64	300	127.7478	125.8152	125.8152	131.4812	131.4010
4	64	300	125.4631	125.3081	125.3081	127.0843	129.3957
5	64	300	127.0310	127.8182	127.8182	127.1430	131.1347
6	64	300	123.8140	124.3480	124.3480	124.3480	127.7725
7	64	300	121.4934	122.8798	122.8798	122.8798	125.6208
8	64	300	127.5587	128.2588	128.2588	121.5776	131.5577
9	64	300	126.5822	127.3955	127.3955	125.7656	130.6337
10	64	300	121.3383	121.3175	121.3175	121.3175	124.2896
11	64	300	121.4095	120.8827	120.8827	123.8298	124.3416
12	64	300	127.2162	127.1965	127.1965	127.1991	131.0166
13	64	300	135.2625	134.3412	134.3412	134.3412	138.4939
14	64	300	134.1782	133.2138	133.2138	133.2158	137.9280
15	64	300	127.2380	127.7343	127.7343	118.6807	131.2456
16	64	300	123.8867	124.2682	124.2682	124.2682	127.6566
17	64	300	126.9284	127.1580	127.1580	128.2636	130.7530
18	64	300	127.3455	127.5601	127.5601	123.0358	132.2074
19	64	300	127.8255	127.9822	127.9822	125.6169	131.5347
20	64	300	125.5412	124.0871	124.0871	126.7370	129.3125
21	64	300	131.2789	132.1340	132.1340	128.4535	133.2765
22	64	300	131.6845	131.3152	131.3152	138.0931	136.5793
23	64	300	137.7149	139.7964	139.7964	138.8284	142.1018
24	64	300	127.1421	127.8106	127.8106	123.4715	130.9123
25	64	300	127.8249	128.3595	128.3595	121.4945	130.8595
26	64	300	135.4506	132.7287	132.7287	132.7288	139.7210

Table C.21: Isolet statistical difference test results (global PCA5)

C	K	NC	TE	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	5	300	3	0	0	0	-	0
2	5	300	3	0	0	0	-	0
3	5	300	1	0	0	0	0	-
4	5	300	3	0	-	0	0	0
5	5	300	3	0	0	-	0	0
6	5	300	3	-	0	0	0	0
7	5	300	3	0	0	0	-	0
8	5	300	3	-	0	0	0	0
9	5	300	3	0	0	0	-	0
10	5	300	1	0	0	0	-	0
11	5	300	1	1	1	1	-	0
12	5	300	3	0	-	0	0	0
13	5	300	1	0	0	0	-	0
14	5	300	3	0	-	0	0	0
15	5	300	1	0	0	0	-	0
16	5	300	3	0	0	0	-	0
17	5	300	1	0	0	0	-	0
18	5	300	3	0	-	0	0	0
19	5	300	3	0	-	0	0	0
20	5	300	3	0	0	0	-	0
21	5	300	3	-	0	0	0	0
22	5	300	1	1	0	0	0	-
23	5	300	3	0	-	0	0	0
24	5	300	1	0	0	0	-	0
25	5	300	3	0	0	0	-	0
26	5	300	3	-	0	0	0	0

Table C.22: Isolet statistical difference test results (global PCA1)

C	K	NC	TE	MLE(g)	MLE	MLL	ML-LOO(f)	ML-LOO(s)
1	64	300	1	1	1	1	-	1
2	64	300	1	-	1	1	1	1
3	64	300	3	0	-	0	0	0
4	64	300	3	0	-	0	0	0
5	64	300	3	-	0	0	0	0
6	64	300	1	-	1	1	1	1
7	64	300	1	-	1	1	1	1
8	64	300	1	1	1	1	-	1
9	64	300	3	1	1	1	-	1
10	64	300	3	0	-	0	0	0
11	64	300	3	0	-	0	0	0
12	64	300	3	0	-	0	0	0
13	64	300	3	0	-	0	0	0
14	64	300	3	0	-	0	0	0
15	64	300	3	1	1	1	-	1
16	64	300	1	-	0	0	0	1
17	64	300	1	-	0	0	0	1
18	64	300	3	1	1	1	-	1
19	64	300	3	0	0	0	-	1
20	64	300	3	0	-	0	0	0
21	64	300	3	1	1	1	-	1
22	64	300	3	0	-	0	0	0
23	64	300	3	-	0	0	0	0
24	64	300	1	1	1	1	-	1
25	64	300	3	1	1	1	-	1
26	64	300	3	0	-	0	0	0

C.4 CONVERGENCE EXAMPLES OF ML ESTIMATORS

In this section we present the global PCA CV and test entropy results of the ML estimators for 10 training iterations. We report on the same classes as summarised in Table 6.23 for comparative purposes.

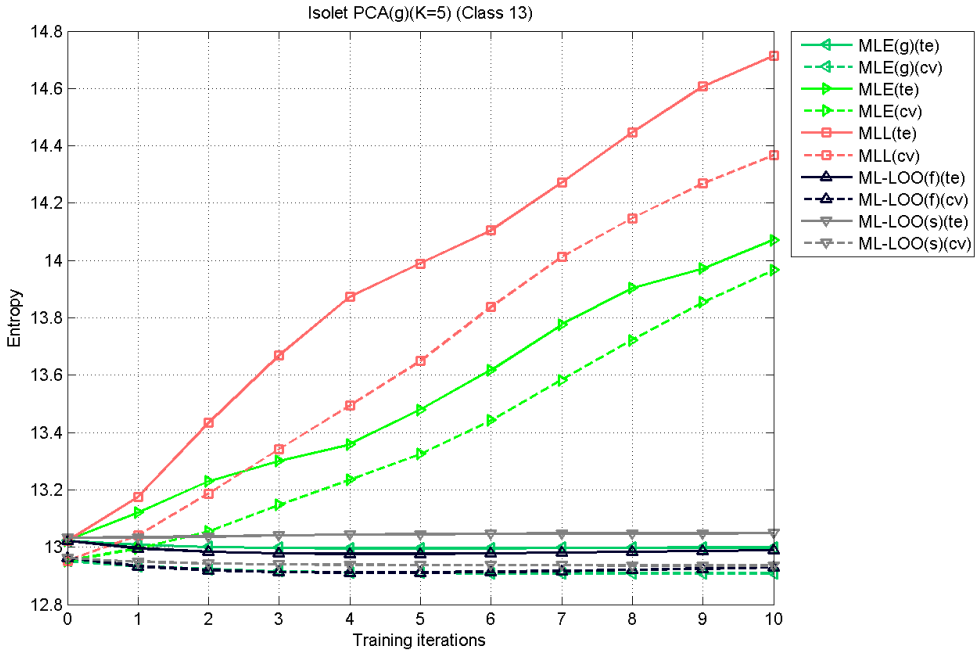

Figure C.1: *Letter entropy results for training iterations (Class 15, Global, PCA5)*

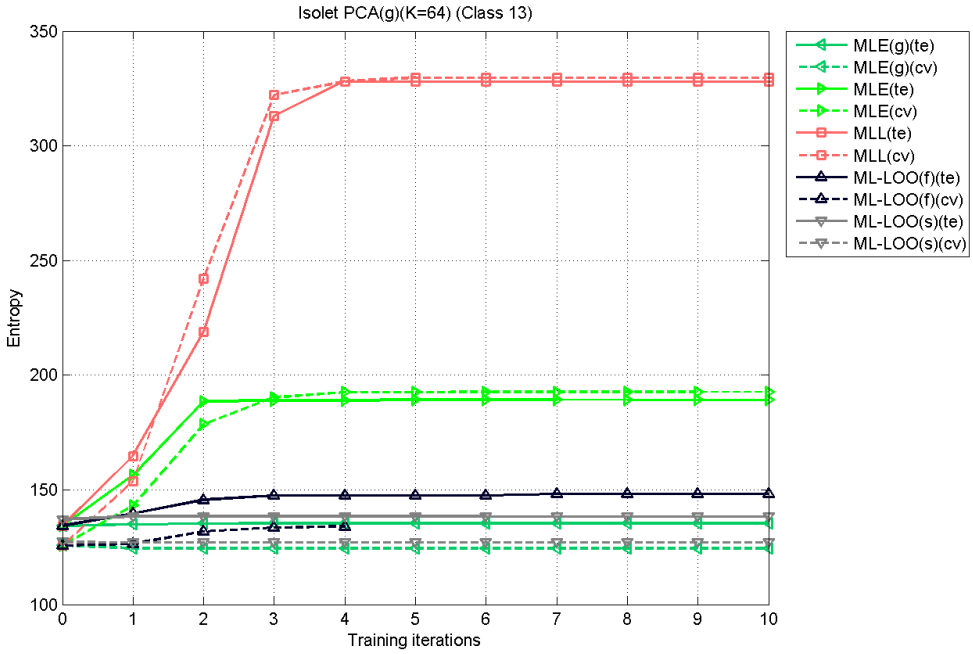

Figure C.2: *Letter entropy results for training iterations (Class 15, Global, PCA1)*

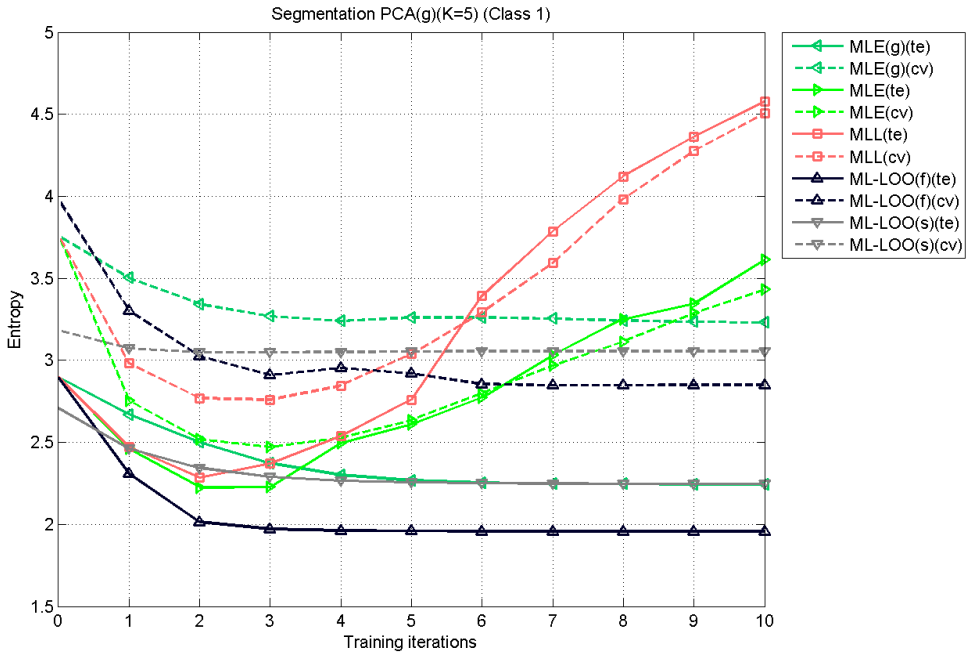

Figure C.3: *Segmentation entropy results for training iterations (Class 1, Global, PCA5)*

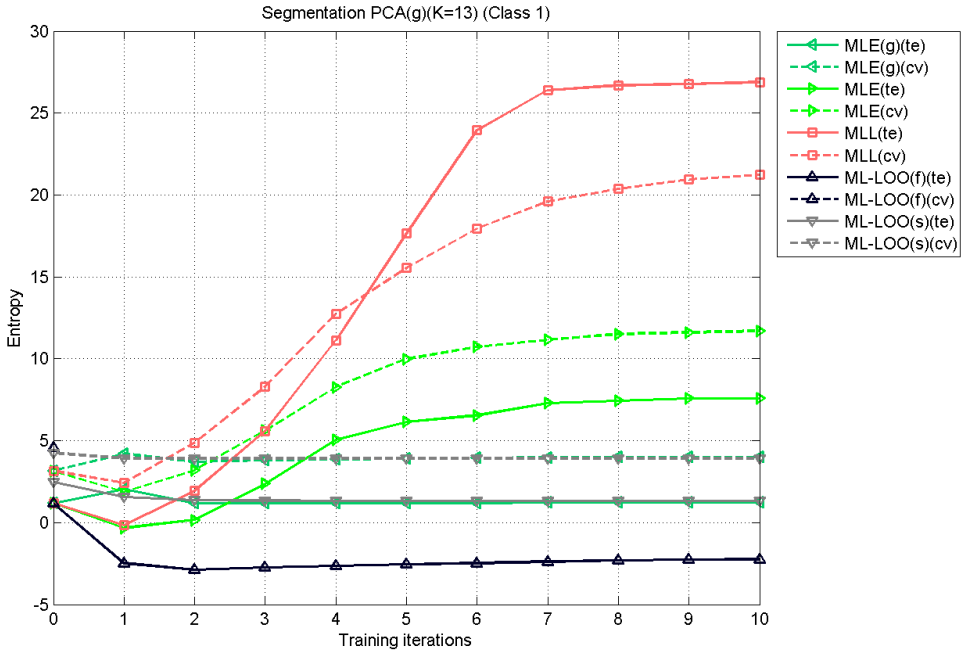

Figure C.4: *Segmentation entropy results for training iterations (Class 1, Global, PCA1)*

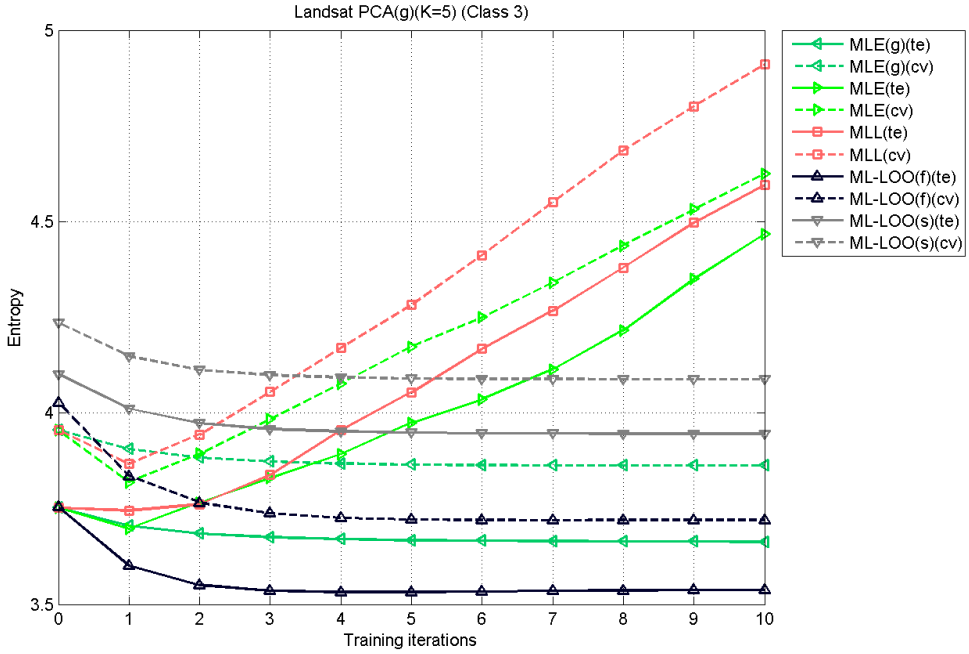

Figure C.5: *Landsat entropy results for training iterations (Class 3, Global, PCA5)*

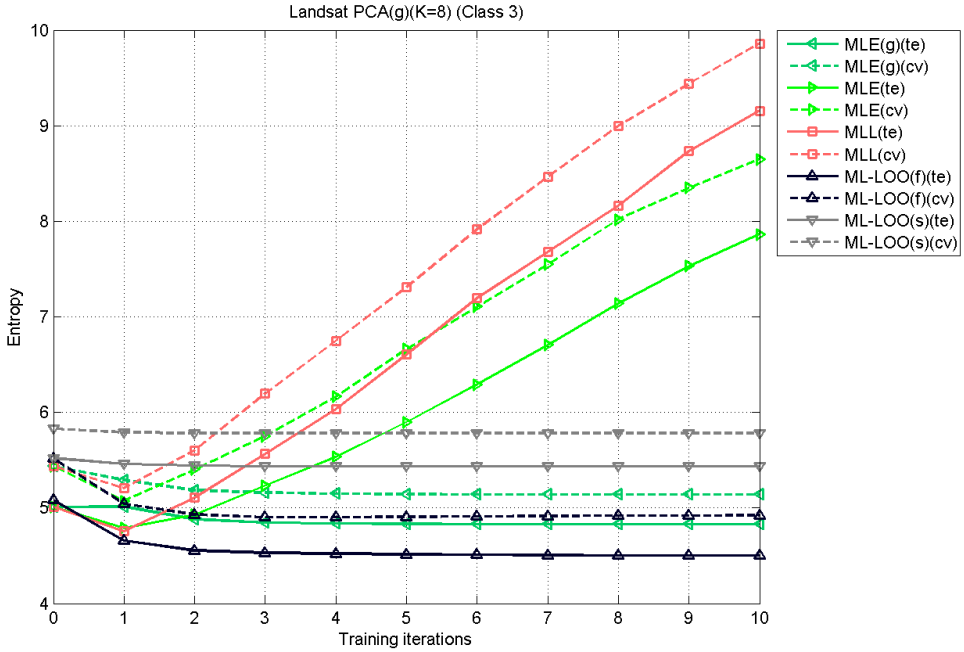

Figure C.6: *Landsat entropy results for training iterations (Class 3, Global, PCA1)*

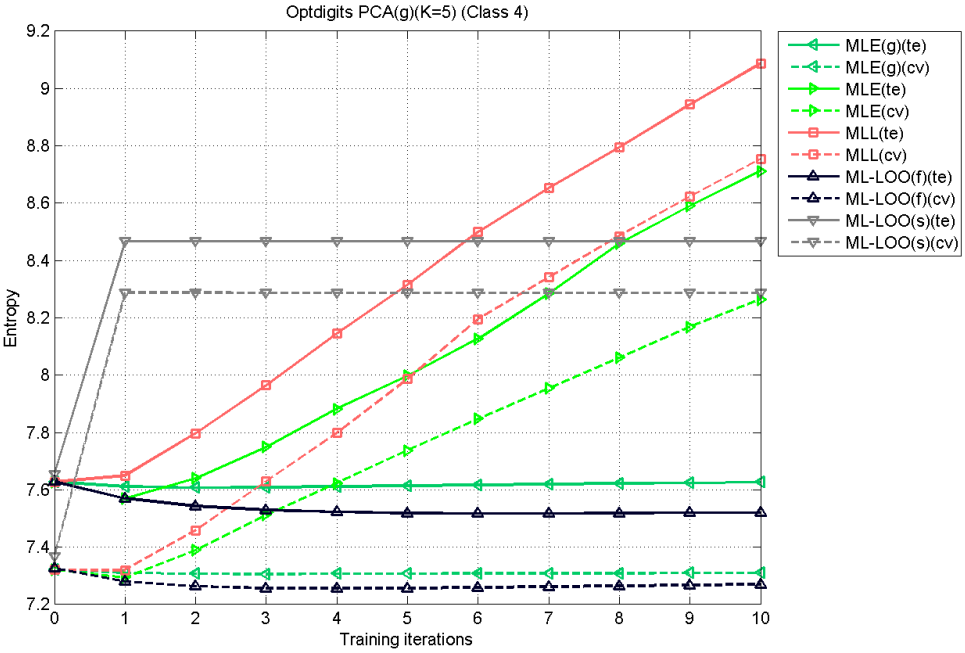

Figure C.7: *Optdigits entropy results for training iterations (Class 4, Global, PCA5)*

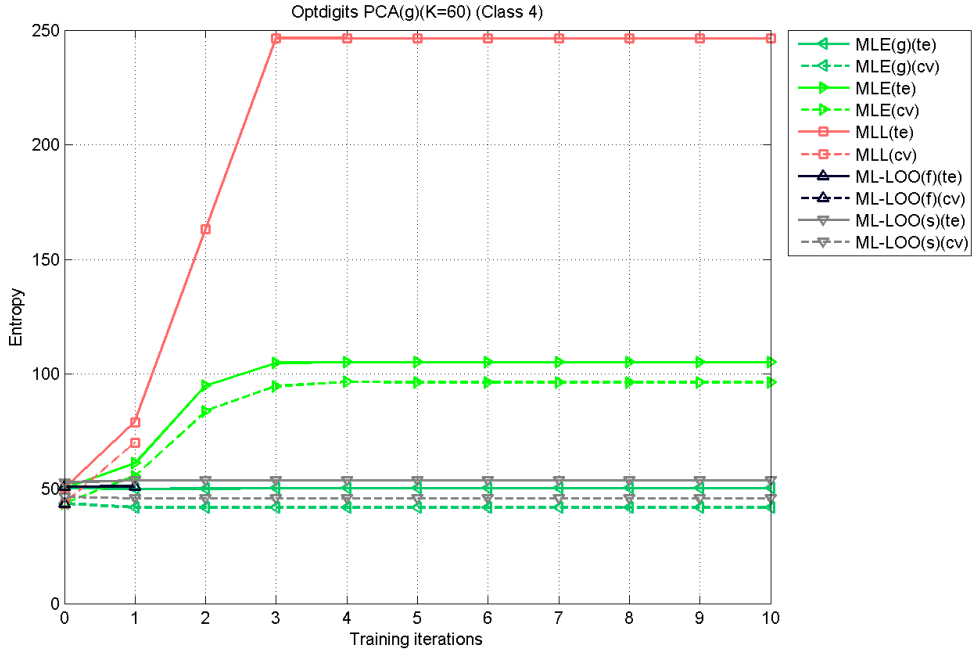

Figure C.8: *Optdigits entropy results for training iterations (Class 4, Global, PCA1)*

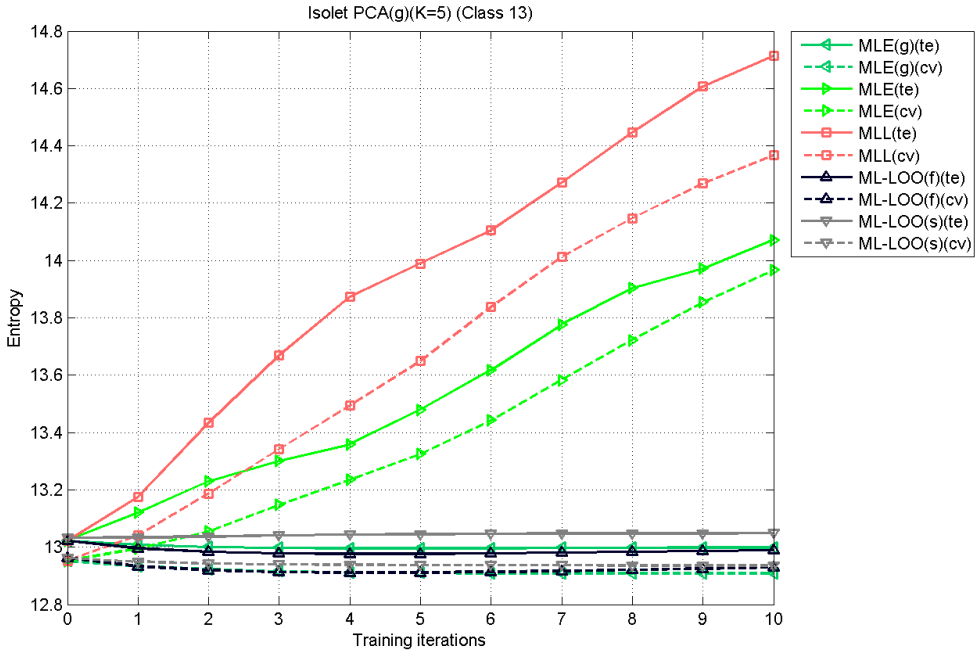

Figure C.9: *Isolet entropy results for training iterations (Class 13, Global, PCA5)*

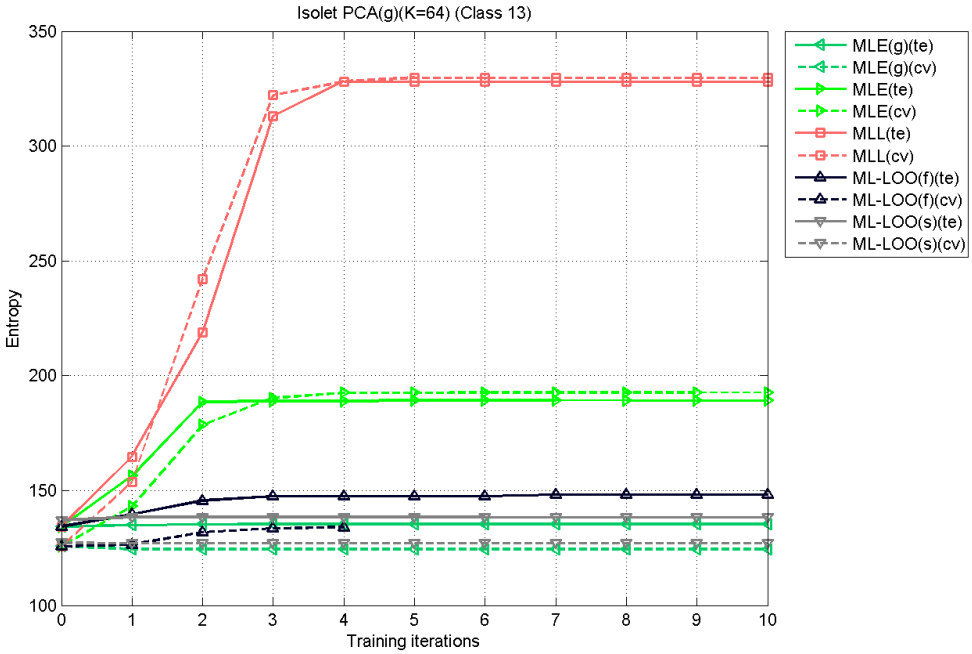

Figure C.10: *Isolet entropy results for training iterations (Class 13, Global, PCA1)*

C.5 CONCLUSION

In Chapter 6 we concluded that for the CV and RW datasets investigated, no estimator clearly performed optimally on all tasks and that the optimal estimator varied between datasets and between classes within the same dataset. We also illustrated with artificial data that the optimal estimator is determined by the properties of the data, such as scale variation between features, scale variation within features, and correlation between features.

In Sections C.2 and C.3 we observe the same relative performance behaviour on the CV and RW datasets for the globally pre-processed PCA data, with some exceptions. On the CV PCA5 data the ML-LOO(f) estimator performs more competitively on the Vehicle and Waveform datasets, while on the CV PCA1 data the ML-LOO(f) estimator performs more competitively on the Balance Scale, Iris, Vehicle and Waveform datasets. On the *Letter* dataset, the MLE(g) experiences a relative improvement in performance on the PCA5 data and the MLL on the PCA1 data. On the *Segmentation* dataset, the MLE and MLL estimators perform more competitively on the PCA5 data, while the ML-LOO(f) estimator experiences a relative degradation in performance on the PCA1 data. On the *Landsat* dataset, the ML-LOO(f) estimator performs more competitively on the PCA5 and PCA1 data. On the *Optdigits* dataset, the ML-LOO(s) estimator

experiences a relative improvement in performance for both the PCA5 and PCA1 data. On the *Isolet* dataset, the ML-LOO(f) estimator performs more competitively on the PCA5 and PCA1 data, while the MLE(g) experiences a relative decrease in performance on the PCA1 data.

Generally the ML-LOO(f) estimator experienced a relative improvement in performance on the globally pre-processed data, compared to the class-specific pre-processed data. This is expected, since the ML-LOO(f) estimator, estimates an identical full-covariance matrix for all kernels. It is thus capable of modelling class-specific correlation. When global pre-processing is performed, the class-specific correlation is not removed as in the case of the class-specific PCA, thus the relative performance of the ML-LOO(f) estimator improves. The relative changes in behaviour of the other estimators are less predictable.

We also concluded in Chapter 6 that for class-specific PCA data, the MLE and MLL estimators converge rapidly, and in many cases converge to the optimal entropy in only a single iteration. We also observed that the MLE and MLL estimators were very susceptible to overtraining and that the test entropy scores drastically increased when the optimal number of training iterations was exceeded. The remaining estimators were not so susceptible to overtraining and converged more slowly to the optimal entropy. The results in Section C.4 show that the same conclusions hold for globally pre-processed PCA data.