BOURNEMOUTH UNIVERSITY

Physically inspired methods and development of data–driven predictive systems

by

Marcin Budka

A thesis submitted in partial fulfilment for
the degree of Doctor of Philosophy

School of Design, Engineering & Computing
Bournemouth University

August 23, 2010

Copyright statement

Abstract

Traditionally building of predictive models is perceived as a combination of both science and art. Although the designer of a predictive system effectively follows a prescribed procedure, his domain knowledge as well as expertise and intuition in the field of machine learning are often irreplaceable. However, in many practical situations it is possible to build well–performing predictive systems by following a rigorous methodology and offsetting not only the lack of domain knowledge but also partial lack of expertise and intuition, by computational power. The generalised predictive model development cycle discussed in this thesis is an example of such methodology, which despite being computationally expensive, has been successfully applied to real–world problems.

The proposed predictive system design cycle is a purely data–driven approach. The quality of data used to build the system is thus of crucial importance. In practice however, the data is rarely perfect. Common problems include missing values, high dimensionality or very limited amount of labelled exemplars. In order to address these issues, this work investigated and exploited inspirations coming from physics. The novel use of well–established physical models in the form of potential fields, has resulted in derivation of a comprehensive Electrostatic Field Classification Framework for supervised and semi–supervised learning from incomplete data.

Although the computational power constantly becomes cheaper and more accessible, it is not infinite. Therefore efficient techniques able to exploit finite amount of predictive information content of the data and limit the computational requirements of the resource–hungry predictive system design procedure are very desirable. In designing such techniques this work once again investigated and exploited inspirations coming from physics. By using an analogy with a set of interacting particles and the resulting Information Theoretic Learning framework, the Density Preserving Sampling technique has been derived. This technique acts as a computationally efficient alternative for cross–validation, which fits well within the proposed methodology.

All methods derived in this thesis have been thoroughly tested on a number of benchmark datasets. The proposed generalised predictive model design cycle has been successfully applied to two real–world environmental problems, in which a comparative study of Density Preserving Sampling and cross–validation has also been performed confirming great potential of the proposed methods.

Contents

Abstract **III**

List of figures **IV**

List of tables **V**

Acknowledgements **VII**

List of abbreviations **IX**

Notation **IX**

1 Introduction **1**
1.1 Background . . . 1
1.2 Project description and goals . . . 3
1.3 Original contributions and publications resulting from this work . . . 4
1.4 Organisation of the thesis . . . 5

2 Machine learning and physically inspired techniques **7**
2.1 Introduction . . . 7
2.2 Designing a predictive system . . . 8
2.2.1 Classical predictive system design cycle . . . 9
2.2.2 Generalised predictive system design cycle . . . 11
2.3 Physically inspired learning . . . 21
2.3.1 Data field models . . . 22
2.3.2 Information Theoretic Learning . . . 25
2.4 Concluding remarks . . . 28

3 Electrostatic Field Classification Framework for incomplete data **30**
3.1 Introduction . . . 30
3.2 Data field classifiers . . . 31
3.2.1 Gravity Field Classifier . . . 31
3.2.2 Electrostatic Field Classifier . . . 32
3.3 Improvements of the original Electrostatic Field Classifier . . . 33
3.3.1 Excess of repelling force . . . 33
3.3.2 Simulation step size . . . 34
3.3.3 Early label assignment and label conflict resolution . . . 35
3.3.4 Curse of dimensionality and distance concentration . . . 36

3.4 The missing data problem . . . 38
3.4.1 Types of missingness . . . 38
3.4.2 Traditional approaches to missingness . . . 39
3.5 Electrostatic field framework for incomplete data . . . 41
3.5.1 Incomplete test data . . . 42
3.5.2 Incomplete training data . . . 43
3.5.3 Incomplete both test and training data . . . 44
3.6 Experiments . . . 47
3.7 Concluding remarks . . . 50

4 Density Preserving Sampling for error estimation and model selection 55
4.1 Introduction . . . 55
4.2 Generalisation error estimation . . . 56
4.2.1 Hold–out and random subsampling . . . 56
4.2.2 Cross–validation . . . 57
4.2.3 Bootstrap . . . 58
4.2.4 Bias and variance of error estimation methods . . . 59
4.3 Information Theoretic Learning for entropy manipulation . . . 59
4.3.1 Renyi's quadratic entropy . . . 59
4.3.2 Auto and cross correntropy . . . 60
4.4 Density Preserving Sampling procedure . . . 61
4.4.1 Estimation of correntropy for unsorted datasets with different cardinalities 62
4.4.2 Correntropy based sampling procedure . . . 63
4.5 Experiments . . . 64
4.5.1 Toy problems . . . 65
4.5.2 Benchmark datasets . . . 68
4.6 Discussion . . . 75
4.7 Concluding remarks . . . 76

5 PDF divergence estimators and their applicability to representative data sampling 77
5.1 Introduction . . . 77
5.2 Estimation of the probability density functions . . . 78
5.2.1 Parzen window method . . . 78
5.2.2 k–Nearest Neighbour method . . . 79
5.3 Probability density function divergence measures . . . 80
5.3.1 Kullback–Leibler divergence . . . 80
5.3.2 Jeffrey's divergence . . . 81
5.3.3 Jensen–Shannon divergence . . . 82
5.3.4 Cauchy–Schwarz divergence . . . 82
5.3.5 Mean Integrated Squared Error . . . 83
5.4 Empirical convergence of the divergence estimators . . . 84
5.4.1 Experiment setup . . . 84
5.4.2 Estimation of the Kullback–Leibler divergence . . . 88
5.4.3 Estimation of the Jeffrey's divergence . . . 88
5.4.4 Estimation of the Jensen–Shannon's divergence . . . 88
5.4.5 Estimation of the Integrated Squared Error . . . 95
5.4.6 Estimation of the Cauchy–Schwarz divergence . . . 95

5.4.7 Summary . . . 95
5.5 PDF divergence guided sampling for error estimation . . . 99
5.5.1 Experiment setup . . . 99
5.5.2 Correlation between divergence estimators and bias . . . 99
5.6 Discussion . . . 105
5.7 Concluding remarks . . . 109

6 Applications **110**
6.1 Introduction . . . 110
6.2 Ridge regression ensemble for toxicity prediction . . . 110
6.2.1 Background . . . 110
6.2.2 Environmental Toxicity Prediction Challenge . . . 111
6.2.3 Data description . . . 111
6.2.4 Main assumptions . . . 111
6.2.5 Generalisation error estimation . . . 112
6.2.6 Base model pool . . . 112
6.2.7 Data preprocessing . . . 114
6.2.8 Ensemble generation and evaluation . . . 116
6.2.9 Experiments . . . 116
6.2.10 Summary of case study findings . . . 117
6.3 Predictive modelling of water pollution using biomarker data . . . 119
6.3.1 Background . . . 119
6.3.2 Dataset properties . . . 120
6.3.3 Classification with a single model . . . 121
6.3.4 Feature selection . . . 126
6.3.5 Ensemble model . . . 130
6.3.6 Experimental results . . . 131
6.3.7 Summary of case study findings . . . 136
6.4 Concluding remarks . . . 137

7 Conclusions **138**
7.1 Project summary . . . 138
7.2 Main findings and contributions . . . 140
7.3 Further research . . . 142

A Benchmark datasets **144**

B Classifiers, regressors and preprocessing techniques **149**
B.1 Classifiers . . . 149
B.2 Regressors . . . 152
B.3 Preprocessors . . . 153

C Information Theoretic Learning framework **155**
C.1 Renyi's quadratic entropy . . . 155
C.2 Mutual information between continuous random variables . . . 156
C.3 Mutual information between continuous and discrete random variables . . . 159

References **160**

List of Figures

1.1 Structure of the thesis and chapter dependencies 6

2.1 Learning strategies and how they relate to each other 8
2.2 Relationship between input, output, and target predictions spaces, true and approximated mappings, noise process and error function 9
2.3 Classical design cycle of a predictive model 10
2.4 Generalised design cycle of a predictive system 12
2.5 Curve fitting example 14
2.6 Errors as a function of the amount of training (a) and performance of models chosen on the basis of (b) training error and (c) test error 16
2.7 Bias–variance dilemma for a two–class classification problem 18
2.8 Bayes error in a two–class classification problem 19
2.9 Relation between entropy and mutual information 27
2.10 ITL criterion for a learning machine 28
2.11 Correspondence between the generalised predictive system design cycle and physically inspired methods described in the thesis 29

3.1 Regularisation coefficient estimation 34
3.2 Early label assignment issue 35
3.3 Distance concentration for various measures, for a random vector drawn from a unit hypercube (solid line denotes the mean value, shaded region denotes the mean $+/-$ 2 standard deviations). Notice the units on the left. All three distances have been compared on the rightmost plot (Euclidean – bottom line, Manhattan – middle line, fractional of order 0.5 – topmost line). 37
3.4 Unit circles for L_2-norm (dotted), L_1-norm (dashed) and $L\frac{1}{2}-$norm (solid) based distance 38
3.5 Overview of the framework architecture 42
3.6 Representation and classification of incomplete test patterns 43
3.7 EFC decision boundaries for a 2–D PCA projection of the Iris dataset for Charge Redistribution (left column) and casewise deletion (right column) as the deficiency level increases. 45
3.8 EFC decision boundaries for a 2–D PCA projection of the Iris dataset for Charge Redistribution (left column) and casewise deletion (right column) as the deficiency level increases (contd.) 46
3.9 Decision boundaries for complete and maximally deficient PCA projection of the Iris dataset 48
3.10 Performance for incomplete test data and various deficiency levels (recognition rates) . . 48
3.11 Performance for incomplete training data and various deficiency levels (recognition rates) 50

3.12 Performance for incomplete both test and training data and various deficiency levels (recognition rates) . . . 51
3.13 Performance for missing labels and various deficiency levels (recognition rates) . . . 52
3.14 Performance for missing labels, incomplete both test and training data and various deficiency levels (recognition rates) . . . 53

4.1 Hold–out method; ‘f1’,‘f2’,‘f3’ denote features, ‘id’ denotes instance indexes . . . 56
4.2 Cross–validation; ‘f1’,‘f2’,‘f3’ denote features, ‘id’ denotes instance indexes . . . 57
4.3 Bootstrap method; ‘f1’,‘f2’,‘f3’ denote features, ‘id’ denotes instance indexes . . . 58
4.4 Density Preserving Sampling procedure; ‘f1’,‘f2’,‘f3’ denote features, ‘id’ denotes instance indexes . . . 64
4.5 Synthetic datasets . . . 65
4.6 Cone–torus – scatter plots of 8 DPS–S folds . . . 66
4.7 Cone–torus – scatter plots of 8 CV folds . . . 66
4.8 Cone–torus – decision boundaries for *qdc* trained on DPS–S (solid line) and CV (dotted line) folds . . . 67
4.9 Synth–mat – decision boundaries for *qdc* trained on DPS–S (solid line) and CV (dotted line) folds . . . 67
4.10 Mean correntropy between each fold and the original dataset . . . 68
4.11 Mean between–fold correntropy . . . 68
4.12 Mean absolute bias (averaged over all classifiers) . . . 69
4.13 Mean absolute bias (averaged over all datasets) . . . 69
4.14 Standard deviation of error estimate (averaged over all classifiers) . . . 70
4.15 Standard deviation of error estimate (averaged over all datasets) . . . 70
4.16 Bias of DPS error estimate calculated using a single fold (averaged over all classifiers) . . . 73
4.17 Correlation between bias and correntropy . . . 73
4.18 Single model v. combination errors for Cone–torus dataset . . . 74
4.19 Single model v. combination errors for Synth–mat dataset . . . 74
4.20 Discrete Error Distributions for Cone–torus dataset and *qdc* (error rates given in brackets) 74
4.21 Discrete Error Distributions for Synth–mat dataset and *treec* (error rates given in brackets) 75

5.1 Contour plots for the toy problems: solid line – $p(\boldsymbol{x})$, dotted line – $q(\boldsymbol{x})$ . . . 85
5.2 Parzen density based Kullback–Leibler divergence estimator $\hat{D}_{KL}$ (horizontal axis – sample size, vertical axis – estimated value) . . . 89
5.3 kNN density based Kullback–Leibler divergence estimator $\tilde{D}_{KL}$ (horizontal axis – sample size, vertical axis – estimated value) . . . 90
5.4 Parzen density based Jeffrey’s divergence estimator $\hat{D}_J$ (horizontal axis – sample size, vertical axis – estimated value) . . . 91
5.5 kNN density based Jeffrey’s divergence estimator $\tilde{D}_J$ (horizontal axis – sample size, vertical axis – estimated value) . . . 92
5.6 Parzen density based Jensen–Shannon’s divergence estimator $\hat{D}_{JS}$ (horizontal axis – sample size, vertical axis – estimated value) . . . 93
5.7 kNN density based Jensen–Shannon’s divergence estimator $\tilde{D}_{JS}$ (horizontal axis – sample size, vertical axis – estimated value) . . . 94
5.8 Parzen density based ISE estimator $\hat{ISE}$ (horizontal axis – sample size, vertical axis – estimated value) . . . 96
5.9 kNN density based ISE estimator $\tilde{ISE}$ (horizontal axis – sample size, vertical axis – estimated value) . . . 97

5.10 Parzen density based Cauchy–Schwarz divergence estimator $\hat{D}_{CS}$ (horizontal axis – sample size, vertical axis – estimated value) 98
5.11 Correlation between bias and divergence estimates averaged over the latter 100
5.12 Correlation coefficient histograms for all dataset/classifier/divergence estimator triplets . 100
5.13 Histograms of datasets and divergence estimators for the 198 constant divergence no–correlation cases (numbers of cases denoted on the vertical axis) 101
5.14 Correlation between signed bias and divergence estimates, averaged over datasets 101
5.15 Correlation map for signed bias and divergence estimates, averaged over classifiers . . . 102
5.16 Correlation maps for each dataset and signed bias (axes like in Figure 5.14) 103
5.17 Correlation coefficient histograms for each divergence estimator and signed bias (X axis: $[-1, +1]$, Y axis: $[0, +40]$) 104
5.18 Histograms of datasets, classifiers and divergence estimators for the 806 high (≥ 0.9) signed bias correlation cases (numbers of cases denoted on the vertical axis) 105
5.19 Scatter plots of the Cone–torus subsamples for lowest divergence values with superimposed decision boundaries of the *qdc* classifier 106
5.20 Scatter plots of the Cone–torus subsamples for highest divergence values with superimposed decision boundaries of the *qdc* classifier 107

6.1 Regression error (MSE) on PCA transformed dataset v. the number of principal components 115
6.2 Predicted v. target values for the blind–test data 118
6.4 Statistical properties of the data 122
6.5 Mean imputation (dashed vertical line) v. imputation from univariate distribution (solid vertical line) for the 10^{th} feature, superimposed on the PDF (blue solid line) 132
6.6 Nested cross–validation scheme. NFOLD denotes the total number of folds. 132
6.7 Test versus validation errors (mean values of repeated cross–validation) 134
6.8 Test versus validation errors (mean values + stdev of repeated cross–validation) 134
6.9 Misclassification rates of the best level 2 models for each test object (NCV) 135
6.10 Feature usage by component classifiers 136

List of Tables

3.1 Dataset details . . . 47
3.2 Performance for incomplete test data and various deficiency levels . . . 49
3.3 Performance for incomplete training data and various deficiency levels . . . 50
3.4 Performance for incomplete both test and training data and various deficiency levels . . . 51
3.5 Performance for missing labels and various deficiency levels . . . 52
3.6 Performance for missing labels, incomplete test and training data and various deficiency levels . . . 53

4.1 Bias and variance summary . . . 70
4.2 Ranking of top 3 classifiers . . . 71
4.3 Correlation between true generalisation error and error estimates . . . 72

5.1 Divergence measure estimators . . . 87
5.2 Automatic bandwidth selection methods . . . 87
5.3 Gaussian kernel covariance matrix . . . 88

6.1 Details of toxicity prediction descriptors and training/test datasets . . . 112
6.2 Regression error on PCA transformed dataset / errors reported in the literature . . . 114
6.3 First–Pass Winners ranking / DPS–U performance . . . 117
6.4 Mussel biomarker data class details . . . 120
6.5 Experiment scenarios . . . 123
6.6 10–fold cross–validation errors . . . 123
6.7 8–fold Density Preserving Sampling errors (DPS–S) . . . 124
6.8 Leave–one–out cross–validation errors . . . 124
6.9 Classification errors after removal of a single feature (10–fold CV) . . . 127
6.10 Classification errors after removal of a single feature (8–fold DPS–S) . . . 127
6.11 Single feature classification errors (10–fold CV) . . . 128
6.12 Single feature classification errors (8–fold DPS–S) . . . 128
6.13 Classification errors for PCA–transformed dataset (10–fold CV) . . . 129
6.14 Classification errors for PCA–transformed dataset (8–fold DPS–S) . . . 129
6.15 Test errors of combinations, component and candidate classifiers (NCV) . . . 133
6.16 Test errors of combinations, component and candidate classifiers (NDPS) . . . 133

A.1 Benchmark datasets summary . . . 145

B.1 Classifiers used in the experiments . . . 151
B.2 Regressors used in the experiments . . . 153
B.3 Preprocessing techniques used in the experiments . . . 154

Acknowledgements

First of all I would like to thank my supervisor Prof. Bogdan Gabrys for inspiring and encouraging me to take on this PhD project. I am also very thankful for his expert advice, invaluable feedback and most importantly, for keeping me motivated at all times. Special thanks also go to Dr. Katarzyna Musiał for quick and efficient proofreading.

The data used in Section 6.3 was collected during the 'Developing an Index of the Quality of the Marine Environment (Marine Environment I.Q.) based on biomarkers' project no. 173343/S40, funded by the Research council of Norway. Thanks are due to the staff at IRIS – Biomiljø for providing the data and collaboration.

Others who contributed to this work are the anonymous reviewers who gave me their valuable feedback to my academic output. Their comments and insights also helped to improve many aspects of this work.

I would also like to thank my wife Dominika for constant support and understanding. Finally, I would like to express my gratitude to my parents for always pushing me in the right direction in both personal and academic sense. This thesis is also dedicated to the memory of those who are no longer with us.

Author’s declaration

The work contained in this thesis is the result of my own investigations and has not been accepted nor concurrently submitted in candidature for any other award.

List of abbreviations

The abbreviations are given in an alphabetical order.

AMISE	Asymptotic Mean Integrated Squared Error
ANN	Artificial Neural Network
BSS	Blind Source Separation
CIP	Cross–Information Potential
CR	Charge Redistribution
CV	Cross–Validation
CWD	Casewise Deletion
DPS	Density Preserving Sampling
DPS–S	Supervised DPS
DPS–U	Unsupervised DPS
DPS–SU	Supervised + Unsupervised DPS
LOO	Leave–One–Out
EFC	Electrostatic Field Classifier
EFCF	Electrostatic Field Classification Framework
EM	Expectation Maximisation
GFMM	General Fuzzy Min–Max
GCF	Generalised Correlation Function
GFC	Gravity Field Classifier
GTU	Generalised Theory of Uncertainty
ICA	Independent Component Analysis
IF	Information Force
IP	Information Potential
ISE	Integrated Squared Error
ITL	Information Theoretic Learning
JIP	Joint Information Potential
kNN	k–Nearest Neighbour
LDA	Linear Discriminant Analysis
LDR	Local Dimensionality Reduction
MAE	Mean Absolute Error
MAR	Missing At Random

MCAR	Missing Completely At Random
MCS	Multiple Classifier System
MI	Mutual Information
MIF	Marginal Information Force
MIMP	Mean Imputation
MIP	Marginal Information Potential
MISE	Mean Integrated Squared Error
ML	Maximum Likelihood
MLP	Multi–Layer Perceptron
MNAR	Missing Not At Random
MMI	Maximum Mutual Information
MRS	Multiple Regressor System
MSE	Mean Squared Error
MV	Majority Voting
NCV	Nested Cross–Validation
NDPS	Nested Density Preserved Sampling
OLS	Ordinary Least Squares (regression)
PC	Pre–Classification
PCA	Principal Component Analysis
PCs	Principal Components
PDF	Probability Density Function
PLS	Partial Least Squares (regression)
PMIP	Partial Marginal Information Potential
PV	Plurality Voting
RoT	Rule of Thumb method
QSAR	Quantitative Structure–Activity Relationship
RBF	Radial Basis Function
RMSE	Root Mean Squared Error
SVM	Support Vector Machine
UCIP	Unnormalised Cross–Information Potential

Notation

symbol	**description**	**example**
x	scalar value	$x = 1$
x_i	scalar value as i^{th} element of a set	$f(x_i) = 2x_i$
$\boldsymbol{x}$	column vector	$\boldsymbol{x} = \begin{bmatrix}1\\2\end{bmatrix}, \boldsymbol{x} = \begin{bmatrix}x_1\\x_2\end{bmatrix}$
$\boldsymbol{x}_i$	column vector as i^{th} element of a set	$f(\boldsymbol{x}_i) = 2\boldsymbol{x}_i$
$\boldsymbol{X}$	matrix	$\boldsymbol{X} = \begin{bmatrix}x_{1,1} & x_{1,2}\\x_{2,1} & x_{2,2}\end{bmatrix}$
$\boldsymbol{X}_i$	vector as i^{th} column of a matrix	$\boldsymbol{X} = \begin{bmatrix}X_1 & X_2 & \dots & X_N\end{bmatrix}$
$\boldsymbol{X}^{[R\times C]}$	matrix with R rows and C columns	$\boldsymbol{X} = \begin{bmatrix}x_{1,1} & x_{1,2} & \dots & x_{1,C}\\ \dots & \dots & \dots & \dots\\ x_{R,1} & \dots & \dots & x_{R,C}\end{bmatrix}$
$\mathcal{X}$	set	$\mathcal{X} = \{x_1, x_2, \dots, x_N\}$
X	random variable or scalar, depending on the context	$\mathcal{D} = \{(X, T)\},\ N = 100$

Since a part of this thesis has been devoted to sampling techniques, for consistency a convention used by statisticians has been followed and the word ‘sample’ denotes a collection of data instances rather than a single instance, which is often the case in the machine learning literature.

”Nature has her own best mode of doing each thing,
and she has somewhere told it plainly,
if we will keep our eyes and ears open.”

Ralph Waldo Emerson (1860), ”Conduct of Life”

Chapter 1

Introduction

The amount of digital data generated worldwide every year is growing at an unprecedented rate. Automatic data acquisition and storage through website tracking, loyalty programmes, industrial process monitoring or medical records to name a few, has never been so cheap and easy. According to the IBM UK predictions, the amount of digital data in the world will double every 11 hours by the end of this year [27].

It might seem that one of the major challenges we are now facing is how to utilise this massive amounts of data to its full potential. There are however other problems, the machine learning community has been trying to address for many years, which concern the quality of acquired data. Unfortunately none of the data acquisition procedures, be it automatic or manual, is perfect in practice. There are many reasons for this. The sensors we are using to monitor various processes have limited precision and sometimes fail, leading to either noisy or outlying measurements, or even no measurements at all. The chemical or biological tests we perform are often mutually exclusive and destructive, leaving us with incomplete information to support further decisions. Finally, human errors made during data collection or while later entering the data into the database also are a source of inconsistences. Moreover, we should always remember that although often used as such, in general the term 'data' is not a synonym of 'information'. Thus there is always a risk that a data collection process, if not planned consciously and carefully enough, can produce a huge amount of useless numbers, resulting in waste of both time and money.

The pursuit of addressing the issues emerging from growing amounts of varying quality data, often stored in different, incompatible formats, left the smart information systems evolving into large number of very specific techniques only capable to work in highly constrained environment, on limited evidence and with poor complexity control mechanisms. This induces the need for some kind of unifying framework, an information–based 'theory of everything' [45] many researchers are constantly looking for.

1.1 Background

With a magnitude of various, constantly evolving predictive modelling techniques in use, the number of completely novel approaches designed from scratch is relatively low. By looking at the last 30 years or so, there was only a handful of truly seminal, ground–breaking inventions in the machine learning field, some dating back even to the mid 20^{th} century.

In the very early days of machine learning, when the computational power was very expensive and in general not easily accessible, the rule based expert systems, sometimes referred to as the 1^{st} generation methods, were a dominating approach [33]. They didn't have high computational

demands, yet in many applications they have performed sufficiently well. A common feature of all rule based systems is, that the decision making process is completely transparent, so it is always possible to explain exactly why the final decision is what it is. This fact in conjunction with reasonable performance in some applications results in expert systems still being used today in a number of cases, like banking or finance planning [79]. Perhaps the biggest drawback of rule based systems is that the rules had to be created manually by a knowledge engineer on the basis of knowledge extracted from domain experts. The knowledge extraction was a daunting process not only because the experts often did not agree with each other, but also because their predictions in many cases tended to be inaccurate. Moreover as the system evolved, the number of rules had to grow, as was the case with the number of exceptions to the rules and then exceptions to exceptions etc. The systems thus started losing their simplicity and transparency. Clearly, some new workhorse of machine learning was needed.

The increase in computational power and its accessibility enabled researchers to look in another direction – learning directly from data and not from the experts anymore. Due to the increasing number of success stories of the Artificial Neural Networks (ANNs) in various applications [11, 69] a huge paradigm shift towards learning from exemplars has been observed, with a central belief that all information needed to develop a predictive model is contained in the data. Domain knowledge has started to be seen as unnecessary, partly because it wasn't easily integrable into the ANN models. The shift to the learning from exemplars paradigm has brought a group of new, previously not known problems, like estimation of the generalisation ability of the developed model [181], the multitude of local minima encountered during model training, which is usually a gradient–driven optimisation process and many more. Nevertheless, the late 80s and the 90s were definitely dominated by the neural networks.

At the beginning of the 21^{st} century the interest has gradually switched towards a new technique – the Vapnik's Support Vector Machines (SVMs) [28, 176, 177], which do not have many of the drawbacks of neural networks and have very sound theoretical foundations. The large number of past and current success stories (the three publications on SVMs referenced above currently have in excess of 45k citations), makes SVMs the state–of–the art technique for classification and regression as of today. SVMs, as well as neural networks are sometimes regarded as the 2^{nd} generation machine learning methods.

Throughout the years there was also a great deal of research effort focused on the application of probability theory to machine learning. The Bayesian artificial intelligence and especially Bayes networks designed and used with the help of graphical models are more and more often seen as the technique for bridging the gap between the domain knowledge driven and data–driven approaches [12, 93]. Although the Bayesian framework effortlessly combines them both, and is especially well suited for online learning, as always, there are some problems. One of the most important is the computational complexity of calculations involving manipulation of possibly high–dimensional probability density functions (PDFs). Since in all but the simplest cases this cannot be done exactly, a number of approximate methods like variational Bayes [12] or expectation propagation [109] has been designed to address this issue. This raises an important question of how much and in what circumstances one can trust the estimates. The reality is that often for the estimates of the probability density functions to be at least remotely accurate, one needs immense amounts of data, which are very computationally costly to process. Nevertheless, the probabilistic Bayesian models certainly have the potential of becoming the 3^{rd} generation machine learning techniques.

All the methods briefly discussed above differ at various levels. One of these differences is the inspiration underlying their development by pushing researchers in previously unexplored

directions. The natural environment is without a doubt the most fruitful source of inspiration in the field of machine learning. At a very high level, observation of a decision making processes performed by the human experts resulted in the development of expert systems. At a lower level, an attempt to copy the principles of operation of human brain led to invention of the ANNs. The multiagent systems [39] or Genetic Algorithms (GA) [51] are just a few more examples.

The mathematical and statistical theories are another rich sources of inspiration, leading to development of the already mentioned Support Vector Machines and the whole field referred to as the Bayesian Artificial Intelligence.

There is however another, often overlooked source of potential inspiration – physics. There exist tremendous similarities between physical world and artificial intelligence in the context of machine learning. At the elementary level of information, uncertainty and complexity matter and information naturally intertwine with each other and the physics of matter and energy often provides the best description of information and its uncertainty [186]. In this setting energy defined as an ability to do work corresponds to uncertainty as an ability to obtain information.

The inherent self–organisation property of various physical systems, which leads to complex emergent behaviour of interacting parts, resulting from following a small set of elementary rules [7] is yet another interesting phenomenon. This property at an individual particle level not only strongly resembles machine learning tasks like clustering or data transformation, but also perfectly fits within the 'learning from exemplars' paradigm mentioned earlier.

Finally, with numerous machine learning problems which still remain open, the guidance of well established and understood physical models may prove extremely beneficial.

1.2 Project description and goals

The main objective of this research project is to explore and investigate some of the similarities between physical world and computational intelligence in order to find inspirations and design a new breed of nature inspired machine learning techniques. The rationale behind this line of research lies in an attempt to adopt the well defined and formalised knowledge describing the physical world to the immature artificial learning field.

This research intends to bridge the gaps between physics and machine learning and provide more efficient and intelligent means for fuller exploitation of evidence which is available with varying quality and in varying quantities. Inspirations for the artificial learning modelling were exploited by a direct application of physical principles describing potential fields to multivariate data vectors treated as charged particles, as well as by using the recently developed Information Theoretic Learning (ITL) framework for online estimation and manipulation of entropy, which also demonstrates deep analogies with physical fields, but without being so strictly constrained.

The study presented in this thesis explores and targets various challenging aspects of the classical predictive system development cycle. This is achieved by exploring usability of strong physical analogies as well as weaker physical inspirations for providing alternative to existing solutions and proposing completely novel ones.

The goals of this research project can be summarised as:

1. To develop and validate on real problems, a predictive system design methodology which would facilitate building of well–performing predictive models, even in the absence of domain knowledge. Application of this purely data–driven methodology to real–world problems should result in predictive systems with performance comparable to the ones designed by the experts in a given problem domain. This will be achieved by offsetting

the lack of domain knowledge by additional computations and automation of all possible activities, which are usually performed manually.

2. To develop a unified classification framework by exploiting the concept of dynamic data–particle based optimisation processes and physical fields interaction, with inherent support for handling deficient inputs and ability to learn not only from labelled data, but also from a mixture of both labelled and unlabelled datasets.

3. To facilitate efficient learning from large datasets, where the computational requirements are currently the biggest obstacle, by addressing this issue at the high level, independent of the actual machine learning model used, taking advantage of the ITL criteria.

The above goals address the predictive system design process at different levels, from individual base models to validation of obtained systems to proposing a novel generalised methodology for building multistage predictive models. As such, they can also be seen as a proof of concept, confirming the potential of physically inspired methods in the field of machine learning.

1.3 Original contributions and publications resulting from this work

The original contributions of this work are:

- Derivation of a rigorous predictive system development methodology/cycle, verified on two real–world problems including the Environmental Toxicity Prediction Challenge CADASTER 2009 [17]. The developed purely data–driven predictive system was awarded as the First–Pass winner, non–significantly different from the top performing submission.

- Comprehensive, extendible Electrostatic Field Classification Framework (EFCF) for supervised and semi–supervised learning from incomplete data [15, 20], which takes advantage of a direct analogy with physical fields by treating multivariate data vectors as charged, interacting particles.

- Novel Density Preserving Sampling (DPS) technique [16, 18] as an alternative to standard, commonly used cross–validation (CV), reducing the computational requirements of generalisation error estimation procedure by an order of magnitude, without compromising the quality of estimate and protecting against typical pitfalls connected with random sampling. DPS is a result of indirect physical inspiration in a form of Information Theoretic Learning framework for online entropy estimation and manipulation.

- Experimental comparative study of probability density function divergence estimators and their usability in sampling for generalisation error estimation [19]. The study calls into question the accuracy of the estimators for datasets smaller than thousands of instances. Nevertheless an attempt is made to exploit the divergence estimators for representative sampling to further reduce the computational requirements of generalisation error estimation by another order of magnitude when compared to 10 times repeated 10–fold cross–validation.

- Multi–stage Multiple Classifier System (MCS) for robust predictive modelling of water pollution using biomarker data [21, 119], developed by taking advantage of the proposed predictive system development methodology and a Missing Not at Random (MNAR) data modelling approach.

The following peer–reviewed conference and journal publications are a result of this work:

- M. Budka and B. Gabrys, "Electrostatic Field Classifier for Deficient Data", in *Computer Recognition Systems 3*, ser. Advances in Soft Computing, M. Kurzynski and M. Wozniak, Eds. Springer, 2009, vol. 57, pp. 311–318.
- M. Budka and B. Gabrys, "Electrostatic field framework for supervised and semi–supervised learning from incomplete data", *Natural Computing*, DOI:10.1007/s11047-010-9182-4.
- M. Budka, B. Gabrys, and E. Ravagnan, "Robust predictive modelling of water pollution using biomarker data", *Water Research*, vol. 44, no. 10, pp. 3294–3308, 2010.
- M. Budka and B. Gabrys, "Ridge regression ensemble for toxicity prediction", *Procedia Computer Science*, vol. 1, no. 1, pp. 193–201, 2010.
- M. Budka and B. Gabrys, "Correntropy–based density–preserving data sampling as an alternative to standard cross–validation", in *Proceedings of the IEEE World Congress on Computational Intelligence*. IEEE, 2010, pp. 1437–1444.
- M. Budka and B. Gabrys, "Density Preserving Sampling (DPS) for error estimation and model selection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010 (Submitted).
- M. Budka and B. Gabrys, "On accuracy of PDF divergence estimators and their applicability to representative data sampling", *IEEE Transactions on Knowledge and Data Engineering*, 2010 (Submitted).
- D. Pampanin, E. Ravagnan, S. Apeland, N. Aarab, B. Godal, S. Westerlund, D. Hjermann, T. Eftestøl, M. Budka, B. Gabrys, A. Viarengo, and J. Barsiene, "The Marine Environment I.Q. concept. Developing an Index of the Quality of the Marine Environment based on biomarkers: integration of pollutant effects on marine organisms." in *Proceedings of the 27th ESCPB (New European Society for Comparative Physiology and Biochemistry) Congress*, 2010 (Accepted).

1.4 Organisation of the thesis

The structure of this thesis and the dependencies between the chapters have been outlined in Figure 1.1. Chapter 2 provides a high level introduction to machine learning and physically inspired techniques. The first part of the chapter focuses on the predictive system development cycle, describing its classical form and then proposing a generalised version of the cycle for data–driven design of predictive systems. In the course of the proposed cycle description, necessary machine learning concepts are progressively introduced whenever a need arises. This way the whole description follows a logical sequence rather than frequently referencing the reader to a dictionary of used terms. The second part of the chapter includes a description of a number of selected physically inspired artificial learning techniques developed to date, and forms the theoretical basis for the rest of this thesis.

In Chapter 3 the first development resulting from this work, in a form of a comprehensive Electrostatic Field Classification Framework is described. An original approach to exploiting incomplete training data with missing features, involving extensive use of electrostatic charge analogy, has been used. The framework supports a hybrid supervised and unsupervised training

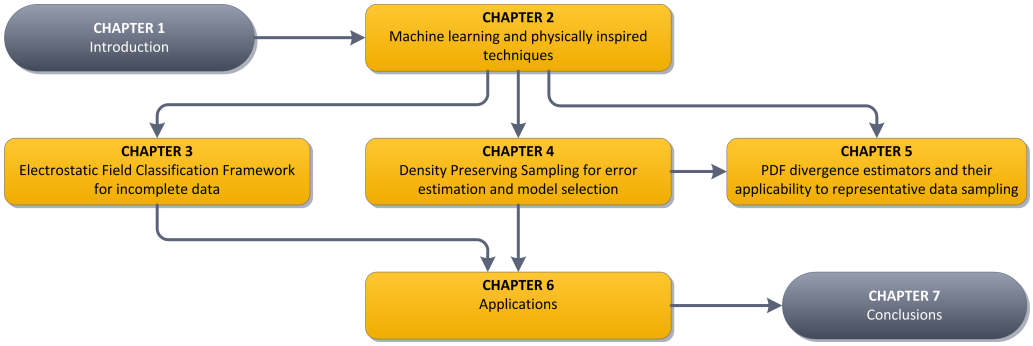


Figure 1.1: Structure of the thesis and chapter dependencies

scenario, enabling learning simultaneously from both labelled and unlabelled data using the same set of rules and adaptation mechanisms. Classification of incomplete patterns has been facilitated by introducing a Local Dimensionality Reduction (LDR) technique, which aims at exploiting all available information using the data 'as is', rather than trying to estimate the missing values. The performance of all proposed methods has been extensively tested in a wide range of missing data scenarios, using a number of standard benchmark datasets in order to make the results comparable with those available in current and future literature. Several modifications to the original Electrostatic Field Classifier (EFC) aiming at improving speed and robustness in higher dimensional spaces have also been introduced and discussed.

Chapter 4 has been devoted to the ITL–based Density Preserving Sampling technique as an alternative to the standard cross–validation, which unlike the latter is not stochastic and thus does not require multiple repetitions in order to produce reliable results, leading to considerable computational savings. DPS divides the available data into subsets by maximising a measure of representativeness of the input dataset. This allows to produce low variance error estimates with accuracy comparable to 10 times repeated cross–validation at a fraction of computations required by CV. The method can also be successfully used for model ranking and selection. The usability and performance of DPS is investigated using a set of publicly available benchmark datasets and standard classifiers.

In Chapter 5 the possibility of selecting a representative subset of data from a larger dataset in a context of accurate estimation of the generalisation performance of a predictive model is further investigated. An experimental comparative study of various estimates of a range of probability density function divergence measures, using a number of synthetic and benchmark datasets is performed. While correlation of the generalisation error with divergence was the primary motivation of this study, it has led to more fundamental analysis of usefulness of the divergence measures, and more accurately their estimators.

Chapter 6 has been devoted to the application of the proposed generalised predictive system development cycle and physically inspired methods to two real–world environmental problems: toxicity prediction and marine pollution monitoring. It is demonstrated that purely data–driven approaches can successfully compete with predictive models developed by the experts in their respective areas. The impact of the physically inspired models derived in this thesis on the two problems is also investigated.

The concluding Chapter 7 summarises the main findings of the project and indicates directions for further research.

Chapter 2

Machine learning and physically inspired techniques

2.1 Introduction

There are many different definitions of machine learning in the literature [4, 12, 40, 110, 162]. At the lowest level of abstraction learning can be described as estimation of the parameter values of a learning machine using a set of available exemplars in order to optimise some criterion function [40]. This definition not only covers all learning scenarios presented in Figure 2.1, but at the same time, in the spirit of the 'learning from exemplars' paradigm, also emphasises the role of training data used during parameter estimation. There are however other ways of looking at the process of machine learning. One of the most general definitions assumes that the learning machine is exposed to a single or multiple sources of information and the goal of learning is to explore and exploit the redundancies from these sources [128].

Whichever definition one chooses to adopt, any method that incorporates information from training data necessarily implies some form of learning. Based on the availability and type of data, the following forms of learning, also presented in Figure 2.1, can be distinguished [40]:

- **Supervised learning** also known as learning with a teacher, in which the learning machine is provided not only with the input data, but also with the corresponding values of the target variable. An example can be a classification task, in which the predictive system is expected to learn how to recognise handwritten characters, and is given both the input data (e.g. a bitmap representing the character) as well as the class label (e.g. information which character the bitmap actually represents). Another example of supervised learning is regression, in which the model learns how to predict values of a continuous (rather than discrete) variable on the basis of available input data.
- **Unsupervised learning** also known as learning without a teacher, in which the learning machine is provided with the input data only. This kind of learning is usually associated with data density estimation or clustering, in which a natural grouping of the input instances is sought. The word 'natural' implies grouping the instances in such a way, that the ones belonging to the same group are similar to each other but dissimilar to the rest. The result of clustering thus strongly depends on the adopted definition of the similarity measure.
- **Semi–supervised learning**, which is a mixture of both supervised and unsupervised learning. It is often the case, that although the supply of labelled data which can be used

for supervised learning is limited for various reasons, unlabelled data is abundant and easy to obtain. Semi–supervised learning algorithms try to take advantage of this fact in order to improve their predictive power and accuracy.

- **Reinforcement learning** also known as learning with a critic, which is an intermediate learning strategy between supervised and unsupervised learning. In reinforcement learning the only information available, apart from the input variables, is if the prediction of the system is right or wrong. The feedback is thus evaluative, not instructive and in extreme situations, it is given only after a long sequence of inputs.

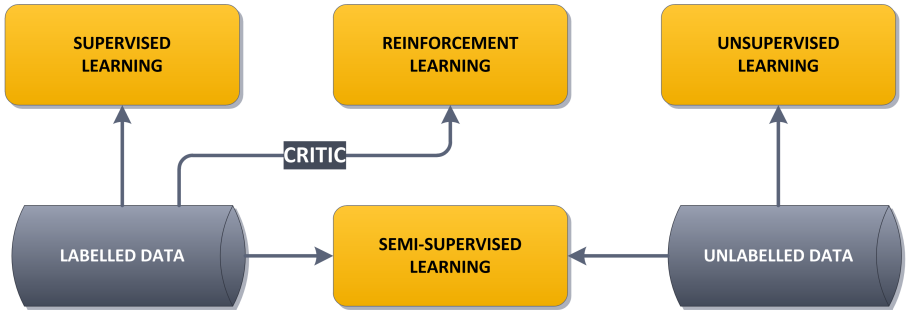


Figure 2.1: Learning strategies and how they relate to each other

2.2 Designing a predictive system

A predictive system S can be defined as a mathematical model, which tries to approximate an existing yet unknown true mapping $M : \mathbb{R}^d \rightarrow \mathbb{R}^c$ from $d-$dimensional input space $\mathcal{X}$ into $c-$dimensional output space $\mathcal{Z}$:

$$M : \mathcal{X} \rightarrow \mathcal{Z} \tag{2.1}$$

For the system S to capture the characteristics of mapping M which are relevant from the point of view of future predictions generated by S, a learning algorithm must be employed in order to adjust the parameters of S in an appropriate way. To achieve this goal, the learning algorithm must also be equipped with sufficient information, which in the spirit of the 'learning from exemplars' paradigm comes in a form of training dataset $\mathcal{D}$ consisting of N instances drawn independently and identically distributed (i.i.d.) according to some probability law $p(\boldsymbol{x})$:

$$\mathcal{D} = \{(X, T)\} = \{(\boldsymbol{x}_1, \boldsymbol{t}_1), (\boldsymbol{x}_2, \boldsymbol{t}_2), \ldots, (\boldsymbol{x}_N, \boldsymbol{t}_N)\} \tag{2.2}$$

where $\boldsymbol{x}_i \in \mathbb{R}^d$ is an input vector and $\boldsymbol{t}_i \in \mathbb{R}^c$ is the corresponding target vector. Denoting by $\boldsymbol{z}_i$ the true value of the mapping M, in general due to limited precision of the measurement instruments and other flaws of the data collection process, $\boldsymbol{t}_i \neq \boldsymbol{z}_i$ but rather:

$$\boldsymbol{t}_i = \boldsymbol{z}_i + \boldsymbol{\epsilon}_i \tag{2.3}$$

where $\boldsymbol{\epsilon}_i$ is assumed to be a zero–mean random noise element [11, 40], that is the expectation $E[\boldsymbol{\epsilon}_i] = 0$. The predictive system $S : \mathbb{R}^d \rightarrow \mathbb{R}^c$ thus becomes an approximate mapping:

$$S : \mathcal{X} \rightarrow \mathcal{Y} \tag{2.4}$$

where $\mathcal{Y}$ is the space of predictions generated by S. It is interesting to note that the above definition applies to both regression and classification problems, as they differ only by the possible values the target variable can take (continuous or discrete).

In order to measure how well a predictive system models the unknown mapping, an error function is used, which in a general form can be written as:

$$err\left(Y,T\right) = \frac{1}{N}\sum_{i}^{N} f\left(\boldsymbol{y}_i, \boldsymbol{t}_i\right) \tag{2.5}$$

where for regression problems typically $f(a,b) = (a-b)^2$ leading to the Mean Squared Error (MSE) function [11, 40] discussed in more detail in the following sections. In the case of classification problems with the target variable being a discrete scalar value representing the class label, often $f(a,b) = (1-\delta_{a,b})$, where $\delta_{a,b}$ is the Kronecker delta function given by:

$$\delta_{a,b} = \begin{cases} 1, & \text{if } a = b \\ 0, & \text{if } a \neq b \end{cases} \tag{2.6}$$

The relationship between input, output, and target predictions spaces, true and approximated mappings, noise process and error function has been depicted in Figure 2.2.

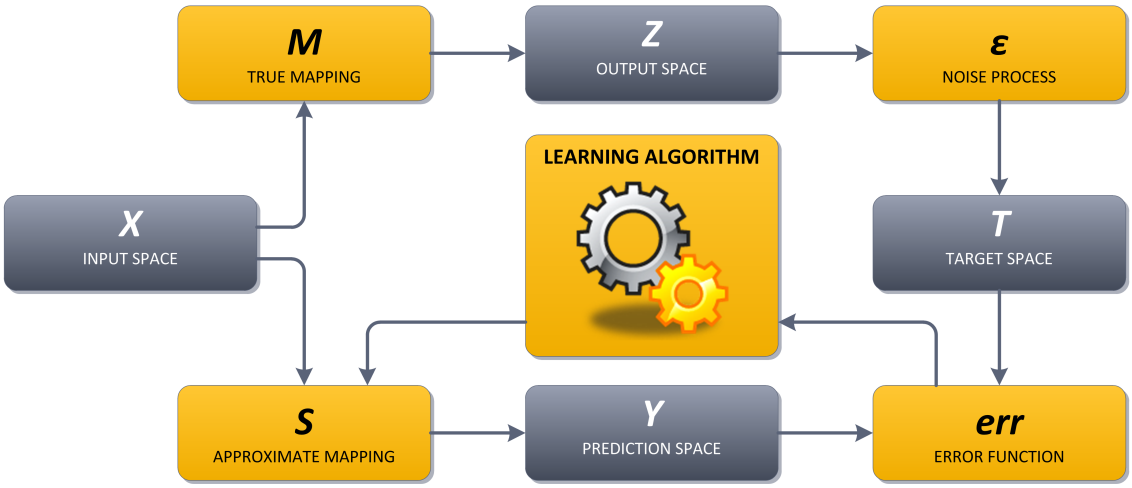


Figure 2.2: Relationship between input, output, and target predictions spaces, true and approximated mappings, noise process and error function

2.2.1 Classical predictive system design cycle

In order to build a well–performing predictive model a systematic approach should be taken, usually leading to an iterative design procedure. A classical predictive system design cycle has been depicted in Figure 2.3 [40]. The cycle entails a number of clearly defined steps, which must be followed and repeated if needed, until a satisfactory result is obtained.

Data collection

The procedure which can account for a large part of time and cost required to develop a system. The main problem at this stage is how to tell that the amount of collected data is sufficient and

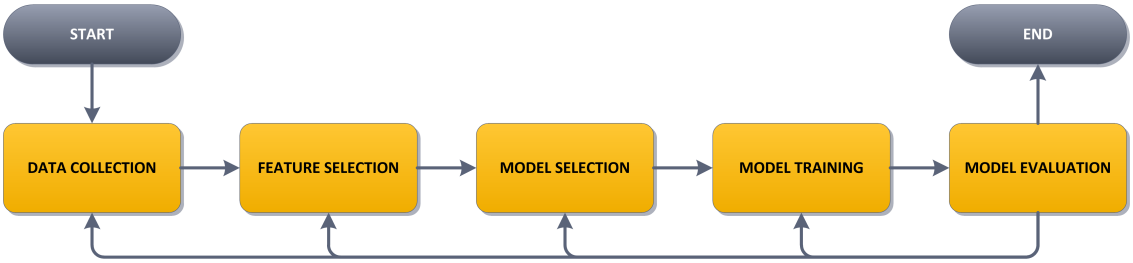


Figure 2.3: Classical design cycle of a predictive model

that the data is representative. In the considerations contained in this thesis it is usually assumed that the data has already been collected and the data acquisition process itself is not investigated, with one exception of the biomarker data used in the study described in Chapter 6.

Feature selection

A critical step strongly depending on the problem, often making use of available prior/domain knowledge, but also possible to be performed in a purely data–driven way. This step involves selection and/or transformation of features/attributes which have been measured during data collection. The selected features should be relevant, simple to extract, insensitive to noise and to irrelevant transformations but it may turn out, that due to a flawed data acquisition procedure none of the features is of any use. The problems which arise here, apart from the obvious 'how to select good feature subset' are often much more subtle, like how not to select wrong features, how to combine domain knowledge with empirical evidence or how many features to choose.

Model selection

With a magnitude of different predictive models to choose from, ranging from simple linear classifiers and regressors, to Artificial Neural Networks and Support Vector Machines, how to choose a correct one for the problem at hand? Or better yet, how to tell if the current model is good or not? One possible solution used throughout this thesis is to select multiple diverse models rather than a single one and combine them to obtain a Multiple Classifier/Regressor System (MCS/MRS) also known as the ensemble model. The problem of scoring the models and discarding the ones which are useless however still remains. Also, if prior knowledge e.g. on suitability of some class of models for similar problems is available it can be used here, but one should not depend on the domain knowledge entirely.

Model training

Training is the process of adjusting model parameters to fit the data. There are many different procedures for training even the same class of models, e.g. the Artificial Neural Networks. For practical considerations, two ANNs trained using different methods are usually treated as two separate models, which shifts the problem of choosing a learning procedure towards model selection. The choice of data for training is also not a trivial issue – should all available data be used to train a model or only some part of it? If only a part of data is to be used, how large should it be, and more importantly, how should it be selected?

Model evaluation

A critical step involving assessment of performance of the designed system and identification of components which need improvements. The central difficulty is what measure of performance to use and how to estimate it. For example, should the error on the training data be used for model evaluation? Or maybe an error on novel, previously unseed data would be a better choice? And if one decides to use the latter, where to get more data? Or perhaps it is possible to do without it?

At this step it is possible to repeat any number of previous steps if model evaluation reveals unsatisfactory performance. In theory thus, the design cycle could be repeated many times, for example using random subsets of features or randomly selected models. In practice however the available computational resources are usually the limiting factor. Does it mean that one should give up randomness completely, although it is known to be useful in many practical cases [11, 40]? Or maybe there is a controlled way to take advantage of the benefits of randomness without stretching the computational resources too much? In order to answer all these questions, a more advanced predictive system methodology is needed.

2.2.2 Generalised predictive system design cycle

A generalised predictive system design cycle, which has been followed in order to develop successful systems described in Chapter 6, is proposed below. Similarly to the classical cycle, it is a number of clearly defined steps one needs to follow to develop a reasonably performing predictive system. The generalised cycle has been depicted in Figure 2.4. The main assumption is that the data has already been acquired, so the data collection step can be omitted. Before describing the proposed design cycle, some of the terms used in Figure 2.4 and the description itself are explained below:

- **Base model pool** is a set of classification or regression models, depending on the problem. For the list of base models used in this thesis please refer to Tables B.1 and B.2.

- **Preprocessor pool** is a set of data preprocessing methods, including but not limited to:

  - feature selection algorithms (e.g. greedy methods – forward, backward, plus–L–takeaway–R [40, 175], semi–random methods – Genetic Algorithms [8, 51] or Simulated Annealing [24, 90] and random methods),
  - linear and non–linear feature transformation (e.g. Principal Component Analysis (PCA) [87], Linear Discriminant Analysis (LDA) [48], Maximum Mutual Information (MMI) projection [9, 166, 167, 168, 169, 170, 165]); some of these methods are discussed in more detail in Chapter 6,
  - outlier detection, denoising and other data cleansing algorithms [178],
  - missing data handling algorithms [63, 117, 136] if required (see Chapter 3 for a more detailed treatment of the missing data problem and a description of some standard protocols for dealing with it).

  Note that preprocessing can have multiple stages, e.g. outlier detection and removal, followed by missing data imputation, followed by Principal Component Analysis. For clarity, in Figure 2.4 and the following description of the generalised design cycle, all these steps would be denoted as a single preprocessing routine.

- **Candidate model** is a base model paired with one or more preprocessing technique, e.g. an Artificial Neural Network using a subset of features (receptive field). In consecutive iterations of the main cycle, whole ensembles from previous iterations can act as candidate models, leading to multistage structures [142].

- **Error estimator pool** is a set of error estimation algorithms, including the hold–out and random subsampling methods [181], the bootstrap method [44], cross–validation [29] and the Density Preserving Sampling technique proposed in Chapter 4.

- **Model selection criterion pool** is a set of criteria for model selection, e.g. select top N models, select models with performance better than average, discard worst 20% of models (also known as trimming) etc.

- **Postprocessor pool** is a set of model combination methods used for ensemble building, including voting combiners (e.g. Majority Voting (MV), Plurality Voting (PV)), averaging combiners (mean, median), weighted versions of the above or even any subset of base models trained on the outputs of ensemble members [139, 173].

The rationale behind using multiple base models, error estimators, pre– and postprocessing techniques is to encourage diversity in the members of the final combined model, as it is believed to have positive stabilising influence on its performance [138]. As a result, a general rule which applies to all the pools listed above is that the larger and more diverse they are, the better. However, in constructing the pools any available domain knowledge should be taken into account, e.g. which base models or preprocessing techniques are better or worse suited for a given problem.

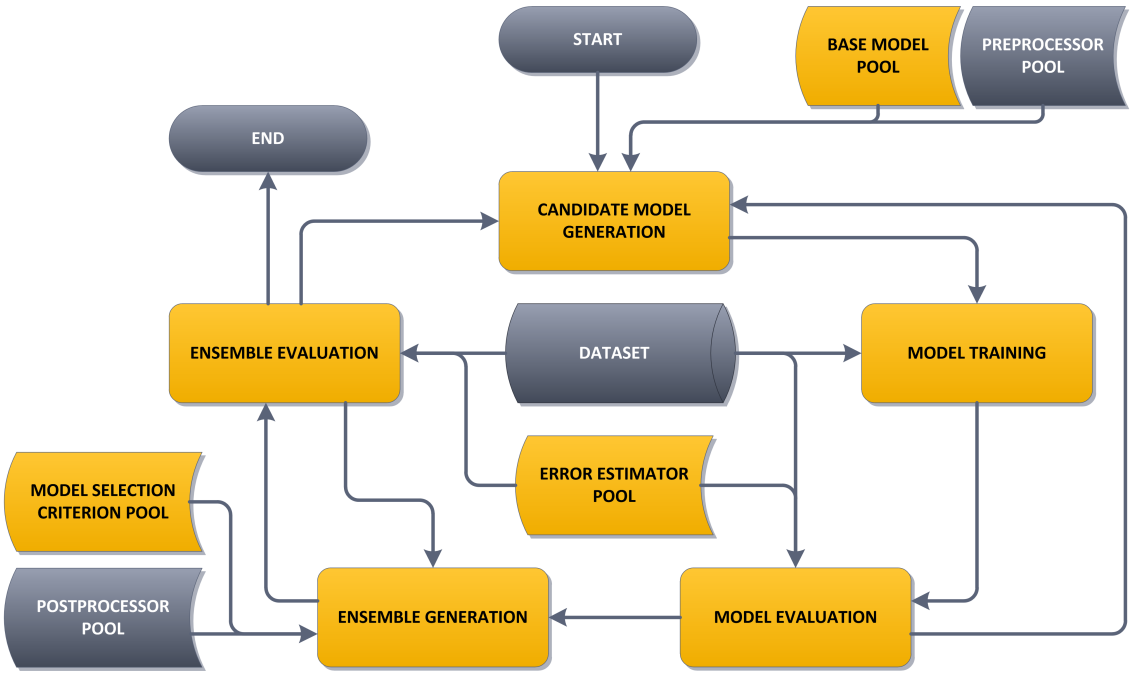


Figure 2.4: Generalised design cycle of a predictive system

Inspection of the data

Inspection of the data is an important activity, which should be treated as a preparatory stage before commencing with building a predictive system. Even superficial manual examination of the data using basic statistical and plotting techniques will quickly reveal any missing or outlying observations, at the same time suggesting what the potential difficulties resulting from the amount, dimensionality and quality of data can be. Inspection of data will often also suggest for example which techniques not to include in the *base model* or *preprocessor pools* (if no data is missing there is no reason to have missing data handling routines in the pool). Although the idea behind the generalised predictive model design procedure is that it can be run autonomously, the findings of this type can save a lot of computations, thus the time spent on data inspection will most likely pay back. Nevertheless due to its manual nature, the step can be treated as optional and has not been explicitly shown on the diagram in Figure 2.4. An example of biomarker data examination using standard statistical techniques is given in Section 6.3.2.

Candidate model generation

In the first step of the generalised predictive system design cycle a given number of candidate models is generated, using various preprocessing techniques and base models. Denoting the chosen base model by M_B and the i^{th} chosen preprocessing technique by $P_{pre}^{(i)}$, a candidate model M_C becomes:

$$M_C = \left(M_B, \left(P_{pre}^{(1)}, \ldots P_{pre}^{(n)}\right)\right) \tag{2.7}$$

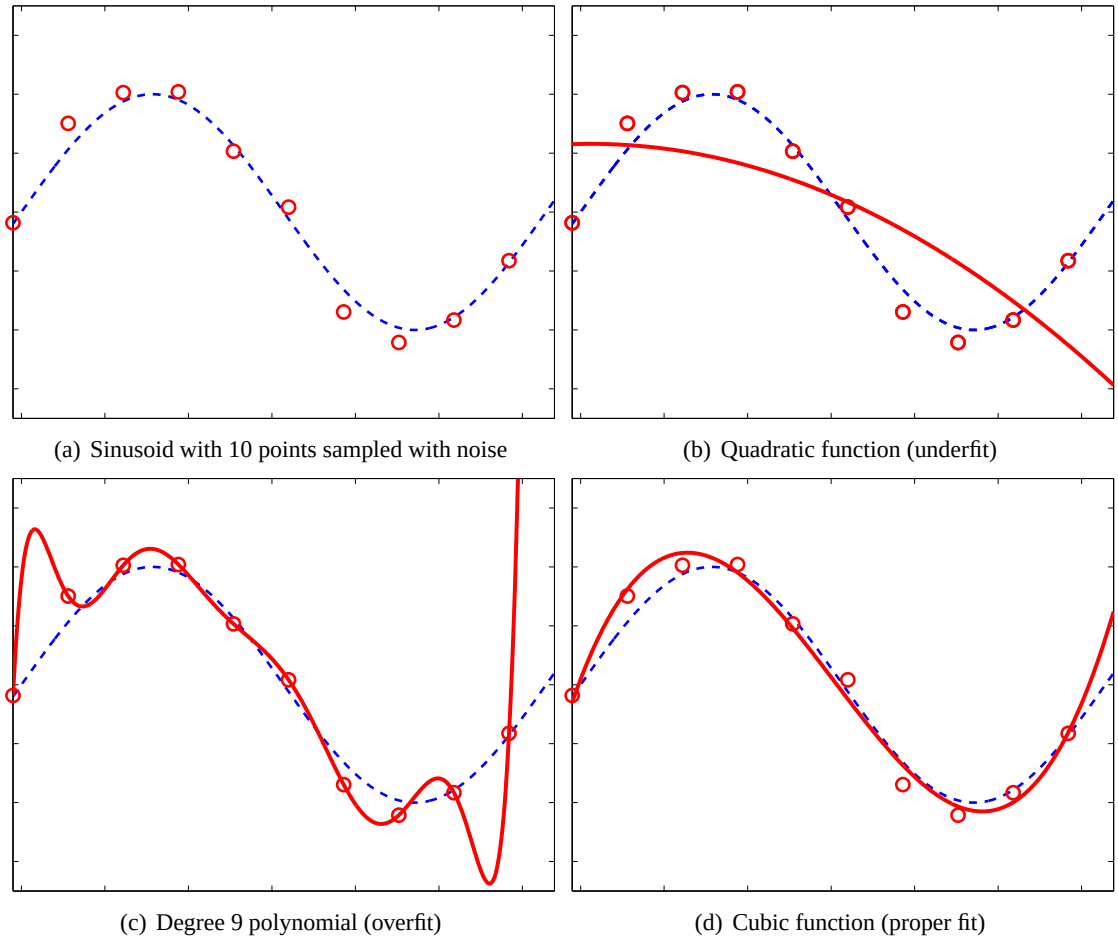
where a single base model can be paired with multiple preprocessing methods to allow for multistage preprocessing as discussed earlier. Generation of candidate models can be achieved by combining preprocessors and base models randomly, using some form of domain knowledge or information gathered during model and ensemble evaluation in consecutive iterations of the main cycle. Even if this kind of information is used, it is important to still allow for some randomness to alleviate the danger of being caught in a locally optimal state, e.g. by using Genetic Optimisation.

The goal of preprocessing is to prepare the data for further use. Apart from addressing the missing data or the outlier problems this way, the usual reason for preprocessing is the dimensionality reduction [11, 12, 40]. The potential problems resulting from working in high–dimensional spaces are collectively known as the 'curse of dimensionality' [11, 40]. Although this phenomenon can have many facets, they all have a common cause – the amount of data, which never seems to be enough and often covers only a fraction of the input space. This is caused by the fact, that the amount of data required to fill the input space grows exponentially with its dimensionality. The solution is to reduce this dimensionality somehow, either by selecting a subset of interesting attributes or performing a transformation into lower dimensions. The dimensionality reduction techniques are also useful from another point of view – they enable detection and removal of collinear and other irrelevant attributes, which might cause numerical problems for some base models. A more detailed treatment of some practical problems resulting from the 'curse of dimensionality' can be found in Chapter 6.

Model training

After the candidate models have been generated they can be trained using available training data. The problem here is that large amount of data with specified values of the target variable is usually difficult and expensive to obtain. Moreover, although it might be tempting to use all available data

for training, this would imply assessing the performance of the obtained predictor on the basis of training error only, in most cases leading to overfitting [11, 40].



(a) Sinusoid with 10 points sampled with noise

(b) Quadratic function (underfit)

(c) Degree 9 polynomial (overfit)

(d) Cubic function (proper fit)

Figure 2.5: Curve fitting example

In order to explain the overfitting issue, following [11] Figure 2.5 presents a simple curve fitting example. The task is to model the unknown underlying function (which is a sine) by using ten points, sampled i.i.d. with noise. This has been depicted in Figure 2.5(a), where it can be seen that most of the points do not lie exactly on the sinusoid, although they follow it closely[1].

In Figure 2.5(b) a quadratic function has been fit to the data and as it can be seen, it does not capture the underlying function nor the training data – high training error would however reveal this fact. The quadratic polynomial is not complex or powerful enough to model the sinusoid and it underfits the data.

In Figure 2.5(c) a degree 9 polynomial fit to the same set of points is presented, capturing the training data perfectly. The underlaying sine function is however not captured very well, especially at the right part of the plot, where the two curves diverge. Unfortunately, since the polynomial passes exactly through all ten training points, the training error is 0, giving a false sense of a success. It is said that the model overfits the data because it is too complex for this particular problem.

[1]There might be various reasons for the measurements to be noisy, usually connected with limited precision of the measurement instrument or process, thus in practice it is reasonable to assume that the data is always noisy.

The best solution appears to be a cubic polynomial, which is neither too complex nor too simple. By examining Figure 2.5(d) it can be seen that the training error is at a reasonably low level (although not 0) and the underlying function is also modelled well (although not perfectly, due to noisy input data). The problem is, that if one was to judge the model basing on the training error only, the complex degree 9 polynomial would be selected, which obviously is a suboptimal choice.

Model evaluation

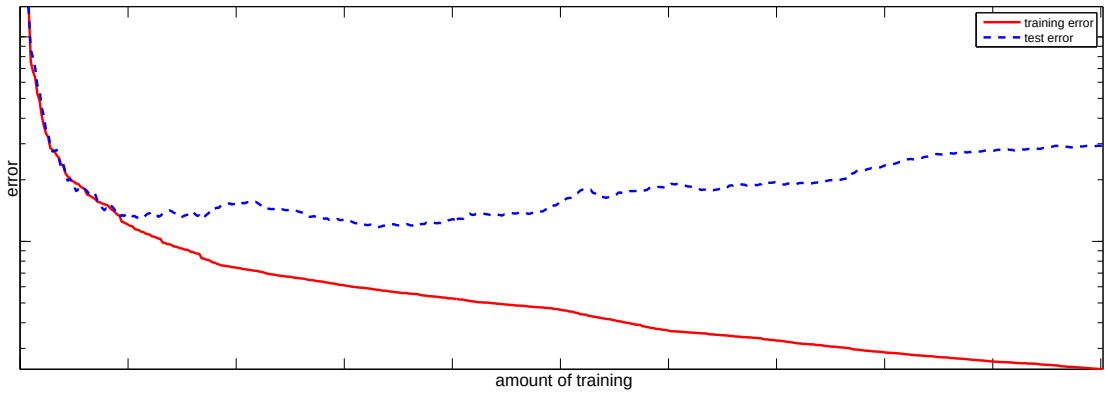
The performance of all trained models is assessed at this step using one or more error estimation methods drawn from the *error estimator pool*. The need for using special error estimation techniques (which should not be confused with the error function given in Eq. 2.5) stems from the behaviour of the predictive models presented in Figure 2.5 and their susceptibility to overfitting. As mentioned before, the training error is usually not a good measure of future performance and the solution is to measure the error using independent test data instead [40].

In Figure 2.6(a) the plot of training and test errors as functions of the amount of training (number of epochs – presentations of the data) for an Artificial Neural Network has been given. For presentation purposes the vertical axis has a logarithmic scale but the actual values on the axes are irrelevant for now. As the number of epochs grows, the training error (solid red curve) becomes smaller and smaller. If the network is powerful enough (that is, if it has enough adjustable parameters), due to its universal approximation property [11] the training error would eventually approach 0. The test error (dashed blue curve in Figure 2.6(a)) however behaves in a completely different way. After initial decrease, it starts climbing and the two curves move apart as the training progresses. This is a clear sign of overfitting – the network begins to 'memorise' the dataset, losing its generalisation ability. It is interesting to note, that for models which are not universal approximators, a plot of errors v. model complexity (degree of the polynomial in the curve fitting example of Figure 2.5) would have a similar shape.

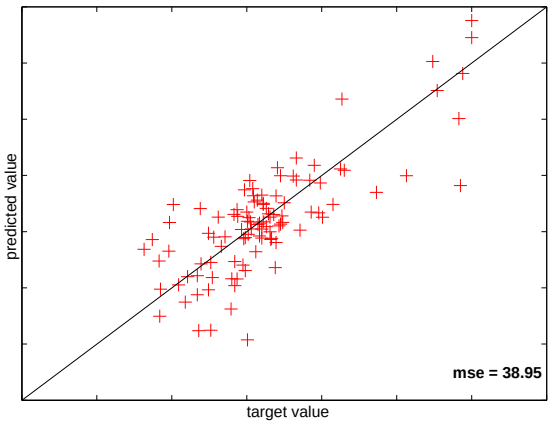
The prediction errors of a network trained until the training error approaches 0 have been shown in Figure 2.6(b). Figure 2.6(c) presents the prediction errors of a network with minimum test error. Both plots have been created using a third, independent dataset. It's easy to see that in case of the second network the points are much more concentred about the $x = y$ line, which denotes better performance (for 0 error all points would lie exactly on the $x = y$ line).

Since additional data for testing is often difficult to get hold of, a common practice is to reserve some part of available data to serve exactly this goal. The question now is, how to divide the data into a training and test part and how big should these parts be, not to risk ending up with training or test data which does not cover large parts of the input space. There exists a number of standard techniques to address this problem, which have already been mentioned in the description of the *error estimator pool*. A more detailed treatment of the error estimation issue can be found in Chapter 4 of this thesis.

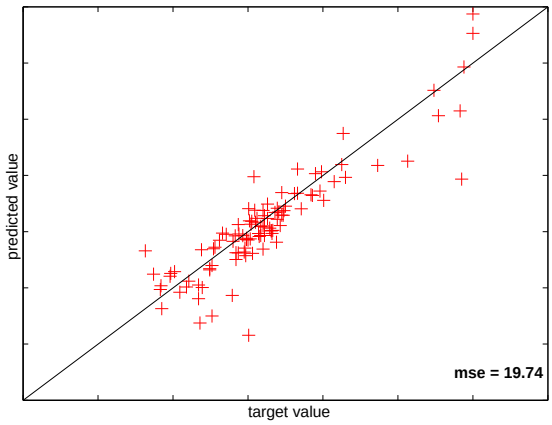
Another important issue which needs to be discussed here is the choice of error function. Traditionally the Mean Squared Error is the most popular choice as it has been a workhorse of machine learning for many years [11]. This is due to its nice analytical properties like continuity, differentiability, existence of a number of effective optimisation algorithms or the so called bias–variance decomposition [57]. MSE however also received some criticism, mainly for not taking advantage of higher order moments of the PDF estimated from data and some ITL based alternatives have been proposed [46, 47, 103, 127, 148]. Using the notation introduced in Eq. 2.5, the Mean Squared Error function can be written as:



(a) Errors v. amount of training for an Artificial Neural Network



(b) Model with the lowest training error



(c) Model with the lowest test error

Figure 2.6: Errors as a function of the amount of training (a) and performance of models chosen on the basis of (b) training error and (c) test error

$$MSE\left(Y,T\right)=\frac{1}{N}\sum_{i}^{N}\left(\boldsymbol{y}_i-\boldsymbol{t}_i\right)^2 \tag{2.8}$$

Since a single value of MSE is dependant on a given realisation of the dataset $\mathcal{D}$, it is interesting to analyse the expectation of MSE considering an average over an infinite number of datasets of size N [11]:

$$E_{\mathcal{D}}\left[MSE\left(Y,T\right)\right]=\frac{1}{N}\sum_{i}^{N}E_{\mathcal{D}}\left[\left(\boldsymbol{y}_i-\boldsymbol{t}_i\right)^2\right] \tag{2.9}$$

After some manipulation and rearranging the expectation inside the sum becomes:

$$E_{\mathcal{D}}\left[\left(\boldsymbol{y}_i-\boldsymbol{t}_i\right)^2\right]=\underbrace{\left(\boldsymbol{z}_i-E_{\mathcal{D}}\left[\boldsymbol{y}_i\right]\right)^2}_{(\text{bias})^2}+\underbrace{E_{\mathcal{D}}\left[\left(\boldsymbol{y}_i-E_{\mathcal{D}}\left[\boldsymbol{y}_i\right]\right)^2\right]}_{\text{variance}}+\underbrace{E_{\mathcal{D}}\left[\boldsymbol{\epsilon}_i^2\right]}_{\text{noise}} \tag{2.10}$$

As it can be seen, the generalisation error of a predictive system can be split into the following three components:

- **Bias term**, which reflects how much on average the mapping S is different from the true

mapping M over all datasets of a given size. If S was a constant function independent of the dataset, although the variance term would vanish, the bias term would most likely be very high, as the data was effectively ignored. This would result in too large values for some datasets and too small for others. In the curve fitting example presented in Figure 2.5, the model based on the quadratic polynomial is highly biased (does not fit the data very well) but has a low variance at the same time.

- **Variance term**, reflecting the sensitivity of the mapping S to a particular realisation of the dataset. If S was a function which fits the training data perfectly (e.g. degree 9 polynomial of Figure 2.5(c)), the bias term would vanish (provided there is no noise) but the variance would likely be high.

- **Noise term**, denoting the inherent noise in the data and at the same time setting a lower bound on the error that can be achieved.

In practice a compromise is sought to provide a good trade–off between fitting the training data and obtaining a smooth mapping able to generalise well, resulting in the so called bias–variance dilemma.

Much of what has been discussed above also applies to the classification problems. Although the error function is usually different (ANNs for classification also often use MSE). The goal is to find a smooth discriminative function, which is not too sensitive to a given realisation of the dataset $\mathcal{D}$ and can thus generalise well.

As mentioned before, in general the difference between the regression and classification problems lies in the type of the target variable. The implications of this fact are however quite important. While in regression the goal is to predict a value of a continuous target variable, in classification the class labels are of interest [40]. As a result, in a $c-$class classification problem the target variable is either a discrete scalar value between 1 and c, denoting membership in one of the c classes $\{\omega_1, \omega_2, \ldots, \omega_c\}$ or a $c-$dimensional binary vector:

$$t_i = \omega_j \;\Leftrightarrow\; \boldsymbol{t}_i = \begin{bmatrix} 0 & \ldots & 0 & 1 & 0 & \ldots & 0 \end{bmatrix}^T \tag{2.11}$$

where the '1' has been placed in the j^{th} row of $\boldsymbol{t}_i$. This is for example the case for ANNs with MSE used as an objective function or any other classifier, which produces soft outputs denoting the degree of membership in each class.

The basic error function used in classification is:

$$err_{clasf}\left(Y, T\right) \;= \frac{1}{N} \sum_{i}^{N} \left(1 - \delta_{y_i, t_i}\right) \tag{2.12}$$

which simply calculates the percentage of misclassified instances. A more elaborate approach also employs a cost function (or cost matrix) stating how expensive each type of an incorrect decision is [40]. For example, in medical diagnostics it would be much more risky and potentially dangerous to classify a person having some serious disease as healthy, than telling a healthy patient that he or she should take some additional tests because something does not seem entirely right. A very simple and often used variant of the cost function is an error weighting scheme based on class prior probabilities [42], which effectively tries to approximate the probability of error. Denoting by $P(\omega_j)$ the prior probability of observing an instance belonging to the j^{th} class,

the probability of error becomes:

$$p_{err}\left(Y,T\right) = \frac{1}{N}\sum_{i}^{N} P(\omega_{t_i})\left(1-\delta_{y_i,t_i}\right) \tag{2.13}$$

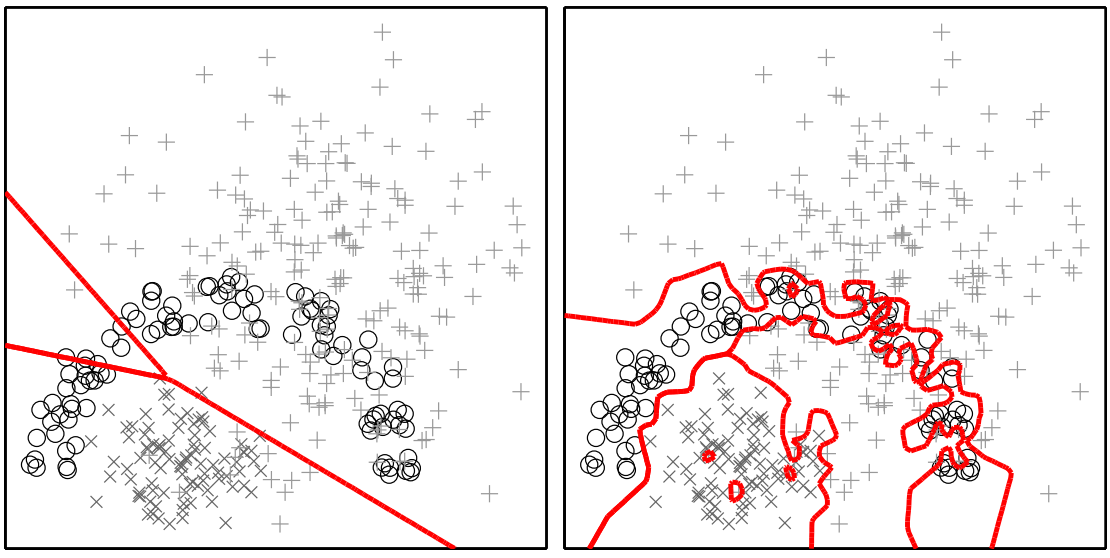
If a classifier produces a soft, probabilistic output, that is:

$$\boldsymbol{y}_i = [p(\omega_1|S,\boldsymbol{x}_i), p(\omega_2|S,\boldsymbol{x}_i), \ldots, p(\omega_c|S,\boldsymbol{x}_i)]^T \tag{2.14}$$

and denoting by ω_T the true class of $\boldsymbol{x}_i$ and by ω_S the class with the highest support given by the classifier, the classification error can be decomposed as [138, 156]:

$$\begin{aligned} err_{clasf}(\boldsymbol{x}_i) &= \underbrace{p(\omega_S|S,\boldsymbol{x}_i)\left[p(\omega_T|\boldsymbol{x}_i) - p(\omega_S|\boldsymbol{x}_i)\right]}_{\text{bias}} \\ &+ \underbrace{\sum_{\omega\neq\omega_S} p(\omega|S,\boldsymbol{x}_i)\left[p(\omega_T|\boldsymbol{x}_i) - p(\omega|\boldsymbol{x}_i)\right]}_{\text{spread}} \\ &+ \underbrace{\left(1-p(\omega_T|\boldsymbol{x}_i)\right)}_{\text{Bayes error}} \end{aligned} \tag{2.15}$$

Similarly to Eq. 2.10, the **bias term** represents the match of the predictive model to the classification problem, while the **spread term** (which is a counterpart of variance) denotes the variability of model outputs from one dataset to another. Figure 2.7(a) presents the decision boundaries of a simple, low variance linear classifier (*fisherc*, for details on the classifier please refer to Table B.1) superimposed on a scatter plot of a synthetic, 3–class Cone–torus dataset (Table A.1).



(a) High bias/low variance linear classifier (b) Low bias/high variance Nearest Neighbour classifier

Figure 2.7: Bias–variance dilemma for a two–class classification problem

As it can be seen, the model does not match the shape of the classes too well (high bias) but the decision boundary is smooth and would not change much if a different dataset sampled from the same distributions was used (low variance).

Figure 2.7(b) depicts an opposite example – a complex, non–parametric Nearest Neighbour classifier (*knnc* in Table B.1), which in fact overfits the data in a pursuit to classify every single instance correctly. This results in high variance component, but due to the fact that the general shapes of the classes have been captured much better than in the case of the linear classifier, the bias component is lower.

The term deserving most attention in Eq. 2.16 is however the Bayes error, which took the place of the noise term in Eq. 2.10 and by analogy denotes the lower bound on the classification error, which is intrinsic to the given classification problem. The name of the error comes from the Bayesian decision theory, in which the Bayes theorem plays a central role [12, 40, 93]:

$$P(\omega_j|\boldsymbol{x}_i) = \frac{p(\boldsymbol{x}_i|\omega_j)P(\omega_j)}{\sum_{k=1}^{c} p(\boldsymbol{x}_i|\omega_k)P(\omega_k)} \tag{2.16}$$

where $p(\boldsymbol{x}_i|\omega_j)$ is the likelihood function, the term in the denominator is a normalising constant called 'evidence' and $P(\omega_j|\boldsymbol{x}_i)$ is the posterior probability of $\boldsymbol{x}_i$ belonging to ω_j. In order to minimise the classification error, for every $\boldsymbol{x}_i$ the class with the highest posterior probability should be chosen.

To illustrate where the Bayes error comes from, Figure 2.8 presents a plot of two likelihood functions specific to two classes ω_1 and ω_2. As it can be seen the problem is that the likelihoods overlap in the shaded region. The dashed vertical line represents an optimal decision boundary minimising the classification error for equal prior probabilities of both classes. In this case, the instances falling into the light–shaded region, whose true class is ω_1 would be erroneously classified as belonging to class ω_2, while in the dark–shaded region the instances whose true class is ω_2 would be erroneously classified as belonging to class ω_1. Thus the total error would be equal to the total shaded area in Figure 2.8, which thus corresponds to the irreducible Bayes error. Note, that if the prior probabilities were not equal, the optimal decision boundary would be shifted right or left in favour of the more probable class, however not affecting the size or shape of the shaded area, resulting in the same value of the Bayes error.

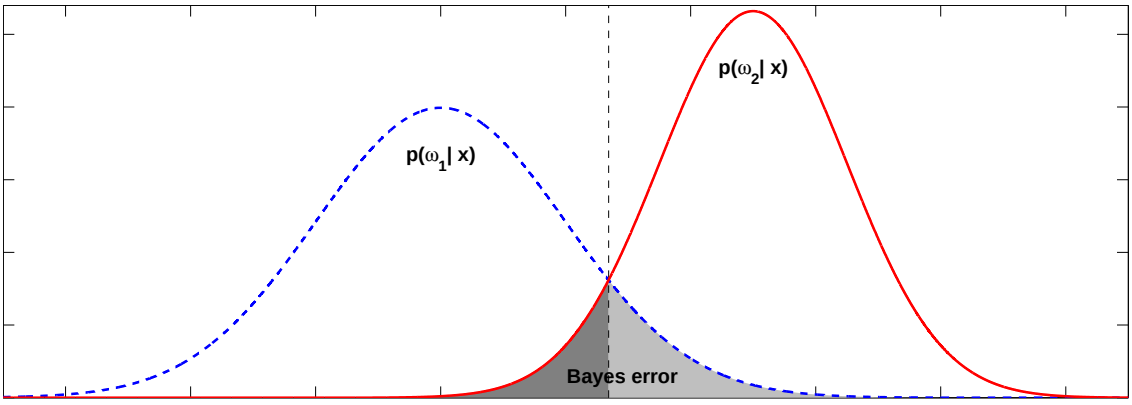


Figure 2.8: Bayes error in a two–class classification problem

All the above considerations must be taken into account during model evaluation, making it in fact an involved and complicated task, which however can be automated to a large extent.

If the evaluation procedure reveals that the performance of the models does not meet some predefined criterion (which can be automatically adjusted as the cycle is repeated), additional candidate models can be generated and trained before progressing to the next step.

Ensemble generation

At this step one or more ensembles are built, using combiners selected from the *postprocessor pool* and evaluated models chosen according to a criterion from the *model selection criterion pool*. The idea behind building ensemble models is to reduce the variance at the expense of computational complexity [98]. Following [36], there are at least three types of reasons supporting the ensemble models:

- **Statistical.** Selection of a single model, which appears to be the best, always carries a risk of making the wrong choice. The situation is even more difficult if there is a number of classifiers or regressors which appear to perform at approximately the same level. Selecting multiple models and combining them is a reasonable solution in this case, as even if the ensemble is not better than its members in terms of the error, the risk of selecting a single bad classifier/regressor is reduced by averaging out any incorrect decisions of a single member. Moreover, the ensemble is more likely to be closer to the optimal yet unknown model, than any randomly picked individual.

- **Computational.** All machine learning algorithms which perform gradient–driven, greedy or random optimisation (e.g. training of ANNs, feature selection using Genetic Optimisation) are susceptible to being caught in a local minimum [40]. This local minimum is however necessarily closer to the global one than at the beginning of the algorithm, which is usually initialised either randomly or heuristically. Aggregation of many models of this kind should thus lead to a better approximation of the optimal model than any single one of them.

- **Representational.** It might be the case that the chosen family of base models is unable to represent the true function which one tries to approximate. For example, a linear classifier is unable to model a non–linear decision boundary. However, a piecewise–linear approximation of this non–linear function in a form of an ensemble of linear classifiers is able to represent it with an arbitrary precision, depending on the size of the ensemble.

The above touches upon a very important problem in the design of combined models – the diversity [99]. The models to be combined should not only be accurate (that is better than random guessing), but they should also be different in order to complement each other, for the combination to be beneficial. After all, having e.g. three opinions on the same problem coming from different sources is better than having one, unless all three are always the same. Unfortunately there is no known universal notion or measure of diversity [138]. As mentioned before, in the generalised predictive model design cycle the diversity is encouraged by using pools of different methods and allowing for some randomness at every possible step of the procedure.

Ensemble evaluation

The performance of the generated ensembles is assessed in this step. Note, that the loop to the previous step not only allows to generate more ensembles if current performance is not satisfactory, but also facilitates building of multistage organisations [142], in which current

ensembles take place of the trained models. A similar structure is developed and described in Chapter 6 to address a real environmental problem.

Note, that in order to reliably assess performance of the ensemble, the data not previously used for training or evaluation of the candidate models should be used. This implies that a third dataset is needed if multiple candidate models are to be combined. One way to address this issue is a nested cross–validation scheme (NCV) proposed in Chapter 6.

If enough computational resources are available, the whole main cycle can now be repeated in order to generate more candidate models and ensembles. In case of postprocessors, which belong to the boosting algorithms group (e.g. AdaBoost [52]), the whole main cycle usually needs to be repeated multiple times, as only one model is included into the ensemble at each iteration.

In practice, development of a well–performing predictive system is in fact a search in a high–dimensional space of base models, pre– and postprocessors, error estimators etc. Although the search is guided, as mentioned before it should allow for some randomness at the same time. Hence as before, the main limiting factor are the available computational resources, which fortunately constantly become cheaper and more accessible.

2.3 Physically inspired learning

It is astonishing how difficult, complex problems are being solved all the time in the physical world, by various biological systems. The ability of animal and human brain to make correct decisions given uncertain, incomplete and noisy evidence, the ease with which social insects like ants, bees, wasps or termites solve complex, optimisation problems are just a few examples. Those phenomena became inspirations for a number of nature inspired learning systems like neural networks, swarm intelligence or genetic and evolutionary algorithms, which have gained popularity in real life applications due to good performance coming at an acceptable computational cost [38, 39, 40, 67]. The ultimate success of those techniques pushed the researchers into a related but somewhat different direction – the physical phenomena.

There exists a number of analogies between the physical world and computing. The oldest one and at the same time the best known is the classical Shannon's definition of entropy [155], which is the basis of the theory of communication and is a counterpart of thermodynamic entropy describing the degree of order among physical gas particles [144, 186].

The Generalised Theory of Uncertainty (GTU) supports an analogy between energy and uncertainty. As opposed to the definition given by the probability theory, GTU distinguishes many forms of uncertainty, perceived as a constraint on the values that a variable can take. In this setting, various forms of energy measured in Joules and seen as capacity to do work, correspond to various forms of uncertainty measured in bits and seen as capacity to obtain information [141, 183, 186].

The Coulomb classifiers described in [76, 77] are another group of physically inspired learning machines. The classifiers form a family of models based on analogy to a system of charged conductors, trained by minimising the Coulomb energy. Three different types of classifiers are proposed, based on uncoupled point charges, coupled point charges and coupled point charges with battery. The Coulomb classifiers are in fact Support Vector Machines but the physical analogy allows to obtain novel types of them comparable or even superior to standard SVMs. The classifiers also provide some insight into the problems that SVM algorithms face in high dimensions caused by using kernels with exponential fall–off, which has stimulated studies on alternative solutions like Coulomb or Plummer kernels.

Yet another example of physically inspired learning was given in [64]. The authors used

an electromagnetic field analogy to derive an efficient, graph based clustering algorithm, outperforming standard clustering techniques especially in high–dimensional spaces.

Some research efforts in the area of physically inspired data mining have also been focused on exploratory data analysis using acoustic data presentation rather than visualisation [70, 71, 80]. The technique termed 'sonification' allows to reveal information about data clustering [71] or principal curves [70] by examining particle dynamics in a data potential. The analysis is based on a model of environmental sound production, assuming on–demand excitation of the system and auditory perception optimised to use the interaction between action and feedback [71].

Finally, the analogy to physical fields resulting in the powerful information field concept [75, 127, 170], which treats data instances as charged particles, allowing each of them to actively contribute to finding the solution of a problem, was exploited in [143].

2.3.1 Data field models

The physical field analogy has been used as a basis of a number of models, resulting in a comprehensive machine learning framework for classification, clustering and data condensation [147]. The basic assumption is that each data instance can be treated as a particle being a source of a central field, which allows for interactions with other particles according to a set of well established rules. The gravity, electrostatic and Lennard–Jones potential field models derived in [147] have been sketched in the following sections. First two of them form a basis of the Electrostatic Field Classification Framework for supervised and semi–supervised learning from incomplete data derived in Chapter 3, thus for their detailed description please refer to Sections 3.2.1 and 3.2.2.

Gravity field model

One of the simplest ways to take advantage of the field analogy in machine learning problems is a model suggested by the gravitation. The gravity field is central and attracting, and is characterised by a negative potential growing with the distance from the field source [65, 143].

The model assumes the field to be static, which means that all data instances being the sources of the field are immobile and fixed to their initial positions. The instances are treated as information particles carrying elementary units of charge and thus each of them is a source of a central field. The superposition of individual contributions of all training instances defines the field potential in any point of the input space. Thus any test instance injected into this field will be subjected to a force, which by analogy to the gravity field, will try to drag this instance into the lowest energy level.

The training dataset uniquely identifies the field, thus the field is given instantly. In a supervised learning scenario this implies that no training process is required.

The gravity field model described above can be used to perform the following tasks:

- **Classification.** The classification procedure with the gravity field model, leading to the Gravity Field Classifier (GFC) is very simple. As mentioned earlier, the potential results in a force able to move a mobile, unlabelled test instance to finally meet one of the fixed field sources and share its label. The magnitudes of the forces are ignored and only their directions are followed, taking a small, fixed step r at a time. This allows to avoid problems in the vicinity of field sources, where the magnitudes of the forces are so big, that the test instances would not only overshoot the sources but would be literally 'ejected' by the field. The whole field is then recalculated and the procedure is repeated

until all test instances approach one of the sources at a distance equal to or lower than r and are labelled accordingly. Since all possible trajectories end up in one of the field sources, the space is thus divided into distinct regions representing classes.

- **Clustering.** Since the labels of training instances in GFC are not used until the very end of the classification process, the mechanism is in fact unsupervised and as such can be almost readily applied to clustering problems. Two modifications are however required: all data instances must be set free to move in the direction of the force, and this movement must be limited by the mass of the instance [143, 144]. Every time two instances come close enough to each other, they are joined to form a new cluster with a mass and charge equal to the sum of joined instances (or clusters). As a result, the interaction of a new cluster with other objects (instances and clusters) becomes stronger, while at the same time the cluster becomes less 'willing' to move. All other calculations are the same as in the case of the gravity field classifier. The granularity level obtained depends on the moment the procedure is stopped [144].

The gravity field model has been described in more detail in Section 3.2.1.

Electrostatic field model

As mentioned before, during the classification process the gravity field model does not take advantage of class labels given with the training data. This information is hence wasted and by exploiting it, one can expect a performance boost. The solution is to employ not only the attracting force but also a repelling force in a way, that instances coming from the same class would attract each other, while instances coming from different classes – repel. This time the physical analogy is the electrostatic field [65].

There is a problem however – the class label of a test instance to be classified is obviously not known in advance (the goal of the classifier is to assign it) and hence it cannot directly interact with the field sources (training instances). To facilitate this interaction, each test instance must be first decomposed into a number of subinstances, belonging to one of the target classes. This can be achieved by using some density estimator (e.g. Parzen window). However, as detailed in Chapter 3, in order not to introduce additional parameters, those partial memberships are assigned in proportion to the gravity field potential of all classes in the test point.

The electrostatic field model described above can be used to perform the following tasks:

- **Classification.** The classification process follows the same rules as in the case of gravity field model. As expected, classification performance and generalisation properties have improved due to better class separation and smoother decision boundaries, which however comes at the price of reduced diversity [143].

- **Data condensation.** The electrostatic field model can also be successfully applied to the problem of data condensation [144]. In this scenario the field sources carrying different class charges keep interacting with each other and moving in the direction of the field forces until no more shifts can be encountered. Similarly to the clustering mechanism, the process becomes a simulation in which the instances are joint together whenever they come into proximity, forming new, heavier data points with combined charges. This merger naturally decreases the total number of instances and can hence be seen as the process of data condensation. Unlike for clustering, the labels of all data points are known in advance and are actively used to influence the field around the instances.

The condensation process follows the procedure described before – at each step the forces are recalculated, the instances are shifted by a small, predefined amount r in the directions of the force vectors, all data points are then tested for mergers and merged accordingly. The simulation can be stopped at any time when the desired condensation level is obtained, or can terminate on its own when only negligible shifts are observed [146, 147].

An important issue is to define how the original class charges evolve during the process and how to resolve conflicts, when two instances from different classes come into proximity. This can be addressed in a number of ways, which resulted in two families of models:

- Crisp Dynamic Data Condensation, where each data point is considered to carry indivisible charge. There are two scenarios possible: unlabelled and labelled. The unlabelled model uses attracting interactions only. Since all instances tend to gradually merge into a single data point, the simulation needs to be applied to each class separately [147]. The result is a set of data points, each of them representing one of the classes. The labelled approach takes advantage of both attracting and repelling forces. Since labels of all instances are known, there is no need for a soft membership assigning function. However, one needs to deal with the issue of collisions, when instances from two different classes meet. The collisions can be resolved using partial class memberships calculated in the same way as in the case of the classification problem to choose the winning class [147].
- Soft Dynamic Data Condensation, where charges are described using some soft membership function. Once again, the partial memberships of the Electrostatic Field Classifier can be used. Density estimates must be however first normalised and scaled to sum up to a unit. There are now three options of how those soft labels can behave during the simulation. The most conservative model (Soft Fixed–Field Condensation) assumes that the field built using the original data is kept unchanged throughout the simulation. Label partitions on the other hand can change freely while an instance travels down the force vector. As a result there are no collisions and the original dataset density structure is preserved as faithfully as possible. This method of condensation significantly outperforms all methods described here in terms of classification accuracy obtained from the condensed set at the same levels of condensation [147]. Two other possibilities are the Soft Fixed–Labels Condensation and Soft Dynamic Data Condensation. The first method keeps label partitions constant while the field is allowed to change freely and the latter does not impose any constraints – both the label partitions and the field can change at each step of the simulation. In both models the class label partitions are summed during mergers [146, 147].

The electrostatic field model has been described in more detail in Section 3.2.2.

Lennard–Jones potential field model

The dynamics of molecules of noble gasses ruled by the Lennard–Jones potential [65] is yet another physical phenomenon, which can be applied to model the interactions between data instances treated as information particles [144, 146]. The field resulting from definition of the Lennard–Jones potential has both attracting and repelling properties. The type of the interaction in this case is however not dependant on the signs of instance charges but on

the distance between them – distant instances are repelled from each other while close ones are attracted to each other.

The Lennard–Jones potential field model can be used to perform clustering in a similar way to the gravity field model.

2.3.2 Information Theoretic Learning

The general definition of machine learning given in Section 2.1 shifts the problem towards quantification and manipulation of the redundancy, which in turn is related to Shannon's information theory [125, 128]. As a result, information theory emerges as the ultimate framework of machine learning. Unfortunately, the application of the information theory to learning problems is not straightforward. The main issue is the omnipresent 'learning from exemplars' paradigm of the learning theory, while the information theory in its traditional form is only able to deal with probability density functions given in the analytic form [128]. Unfortunately in most machine learning problems this form is rarely known and good approximation of the true PDF with a parametric density model is often impossible [40]. Having these issues in mind, an Information Theoretic Learning framework for both non–parametric entropy estimation and manipulation has been derived, enabling training of both linear and nonlinear mappers [128].

Information Theoretic Learning is a procedure of adapting the parameter values of a learning machine using information theoretic criterion [125, 128]. As the goal of learning is to transfer as much information as possible from the training data into parameters of the system, ITL appears to be an appropriate and natural choice. There are basically two main criteria for ITL, both operating in the output space of the predictive system:

- **Entropy**, which is a function of one random variable. As a result, entropy manipulation in a natural way fits the unsupervised learning scenario, although it can also be extended to supervised learning by manipulating the entropy of error [46]. By maximising entropy, ITL can be applied to problems like Principal Component Analysis, whereas entropy minimisation can be used for redundancy reduction or classification [59, 83, 128]. One of the developments resulting from application of the entropy criterion to time series forecasting is a localized, probabilistic measure of similarity between two PDFs [102, 103, 104, 148], which has been used in Chapter 4 of this thesis in order to develop the Density Preserving Sampling scheme.

- **Mutual information (MI)**, which is a function of two random variables and relies on a divergence measure between their joint PDF and product of their marginal PDFs. The choice of those variables determines if the learning is supervised or not. Similarly to entropy, mutual information can also be maximised or minimised and can be used for a wide range of applications like Independent Component Analysis (ICA), BSS, clustering [128] or feature extraction [9, 166, 167, 168, 169, 170, 165].

Entropy criterion

To emphasise the fact that the manipulation of the ITL criterion takes place in the output space of the learning machine, the definitions given below assume Y as the default variable rather than typically used X. Denoting by $P(Y_k)$ the probability of a discrete event Y_k, the Hartley

Information Measure is given by the following formula [30]:

$$H(Y_k) = \log \frac{1}{P(Y_k)} \tag{2.17}$$

Shannon's entropy [155] is simply the expectation of H, which denoting the number of all possible discrete events by N, is given by:

$$H_S(Y) = \sum_{k=1}^{N} P(Y_k)H(Y_k) = \sum_{k=1}^{N} P(Y_k) \log \frac{1}{P(Y_k)} = -\sum_{k=1}^{N} P(Y_k) \log P(Y_k) \tag{2.18}$$

Entropy is a measure of the uncertainty associated with a random variable. Put another way, entropy quantifies the average information content that is missing due to the unknown value of the random variable. As a result, the higher the entropy, the more information can be gained by discovering the value of the random variable.

The extension of Shannon's Entropy to continuous random variables (differential entropy) is:

$$H_S(Y) = \int p(y) \log \frac{1}{p(y)}\, dy = -\int p(y) \log p(y)\, dy \tag{2.19}$$

Switching to the entropy criterion allows to exploit higher order statistics of the distribution, like skewness (degree of symmetry – third moment) or kurtosis (relative peakedness/flatness – fourth moment), which are not taken into account at all by the commonly used Mean Squared Error criterion. This in turn lifts the usual 'Gaussianity' assumption (Gaussian distribution is completely described by the first two moments – mean and (co)variance), which does not hold in many real–world problems and although greatly simplifies the calculations needed to solve them, often distorts the results at the same time [40].

A more detailed treatment of entropy and its estimation has been given in Section 4.3.1 as it forms a basis for the Density Preserving Sampling procedure described in Chapter 4.

Mutual information criterion

Mutual information is defined as a quantity that measures the dependence of two random variables or the amount of information that one variable carries about the other. The classical definition is given as [30]:

$$I_S(X, Y) = H_S(X) - H_S(X|Y) = H_S(Y) - H_S(Y|X) \tag{2.20}$$

where $H_S(X|Y)$ is the Shannon conditional entropy, which denotes the remaining uncertainty of X, when the value of Y is known. I_S hence quantifies the amount by which the uncertainty about X diminishes after observing Y [170]. The relationships between entropy, conditional entropy and mutual information have been presented in Figure 2.9.

Mutual information for two discrete random variables can be calculated using the following formula, which is in fact equivalent to Eq. 2.20 after substituting the definitions of entropy and conditional entropy:

$$I_S(X, Y) = \sum_{k=1}^{N_X} \sum_{l=1}^{N_Y} P(X_k, Y_l) \log \frac{P(X_k, Y_l)}{P(X_k)\, P(Y_l)} \tag{2.21}$$

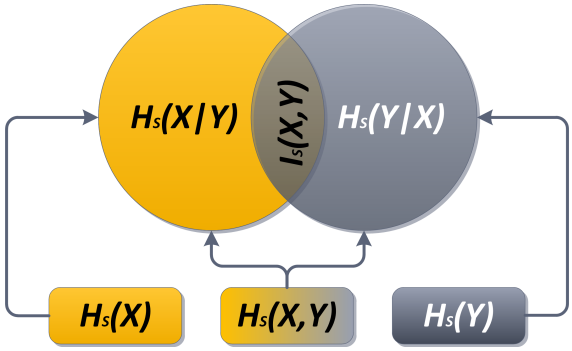


Figure 2.9: Relation between entropy and mutual information

Extension of the above formula to the continuous variable case is straightforward:

$$I_S(X,Y) = \iint p(x,y) \log \frac{p(x,y)}{p(x)p(y)}\, dx\, dy \tag{2.22}$$

Eq. 2.22 is actually a special case of the Kullback–Leibler divergence (discussed in more detail in Chapter 5), which is the most commonly used measure of similarity between two distributions, even though it is not a true distance as it is not symmetric [40]. The Kullback–Leibler divergence between two PDFs $f(z)$ and $g(z)$ is defined as:

$$D_{KL}(f(z), g(z)) = \int f(z) \log \frac{f(z)}{g(z)}\, dz \tag{2.23}$$

When $f(z) = p(x,y)$ is a joint probability density function of two random variables and $g(z) = p(x)p(y)$ is a product of the marginal PDFs, D_{KL} is exactly equal to the mutual information [169]. As a result, MI can be thought of as a distance measure between joint density and a product of marginals, since it is always non–negative and symmetric. Note, that MI is equal to zero only if the variables are independent, that is $p(x,y) = p(x)p(y)$ as in this situation the logarithm in Eq. 2.22 is driven to zero. The above properties can be used for deriving some easily computable estimates of MI. For a detailed description of various mutual information estimation methods please refer to Appendix C.

Mutual information can be used in both supervised and unsupervised learning scenario – all possible learning schemes have been depicted in Figure 2.10. There are 3 signal sources to choose from, represented by the switch 'SW' in one of three possible positions [128]:

- Position 1 implies unsupervised learning since no external desired (target) signal enters the system. The MI criterion is evaluated using only system outputs. This setup can for example be applied to the Independent Component Analysis problem by minimising the mutual information to make the outputs as independent as possible.

- Position 2 also implies unsupervised learning, this time though, the criterion is evaluated using system input and output. By maximising MI in this scenario the system is trained to transmit as much information between input and output as possible.

- Position 3 represents supervised learning as the desired signal is external to the system. If the desired response is a class label (discrete variable), maximisation of MI facilitates feature extraction for the purpose of classification. If the response is a continuous function,

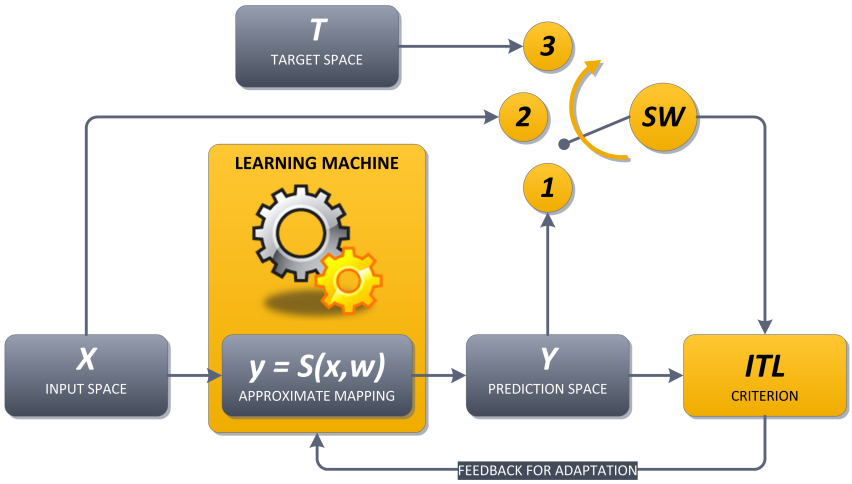


Figure 2.10: ITL criterion for a learning machine

the input is projected to best approximate the desired signal (in the information sense).

As it can be seen, the information theoretic criteria appear as a very attractive alternative to the ones commonly used, including the 'workhorse' of machine learning – MSE. Yet, although the foundations of information theory have been laid down by Shannon in the middle of the 20^{th} century, the number of their to date applications is rather small. The reason for this is the difficulty of estimating entropy and mutual information directly from data [128], since as already mentioned, the formulas given by Shannon require analytical forms of the PDFs.

It is worth noting, that Shannon's entropy and mutual information are not the only measurers relying on the PDFs. Chapter 5 of this thesis presents a study of sampling using various PDF divergence measures, most of which also require the probability density functions to be given analytically. Since this is rarely possible, various PDF approximators are routinely used. As shown later, this can lead to a rather poor estimation of the divergence from data, which can cause numerous difficulties.

2.4 Concluding remarks

In order to design a well–performing predictive system a conscious, methodological approach is needed. The generalised predictive system design cycle presented in Section 2.2.2 is an example of such methodology, which has been successfully applied to real–world environmental problems described in Chapter 6. The proposed procedure has been designed to be run autonomously, in a purely data–driven way, offsetting the lack of expert domain knowledge by computational resources and automation of many steps, which are usually performed manually. However, integration of this kind of knowledge, even if it results from nothing more than basic statistical analysis of the data before building the predictive system, can save massive amounts of computations. Taking into account that the available computational resources usually are the limiting factor, this can lead in effect to better performing systems, as a good solution can obviously be found more easily in a smaller constrained space.

In the studies presented in further parts of this work the following physically inspired techniques discussed in Section 2.3 have been used:

1. **The gravity and electrostatic field models** for development of a comprehensive,

extendible Electrostatic Field Classification Framework for supervised and semi–supervised learning from incomplete data (Chapter 3). The framework fits within the **base classifier and preprocessor level** of the proposed generalised predictive system design cycle.

2. **The Information Theoretic Learning framework** for:

   - Development of the Density Preserving Sampling technique as an alternative to standard cross–validation, reducing the computational requirements of generalisation error estimation procedure (Chapter 4). DPS fits within the **error estimator level** of the proposed generalised predictive system design cycle.
   - A study of PDF divergence estimators and their application to sampling, in order to further reduce the computational requirements of repeated CV, where the ITL–based Cauchy–Schwarz divergence measure was expected to play a central role.

The correspondence between selected steps of the generalised predictive system design cycle and physically inspired methods investigated in this thesis, has been presented in Figure 2.11.

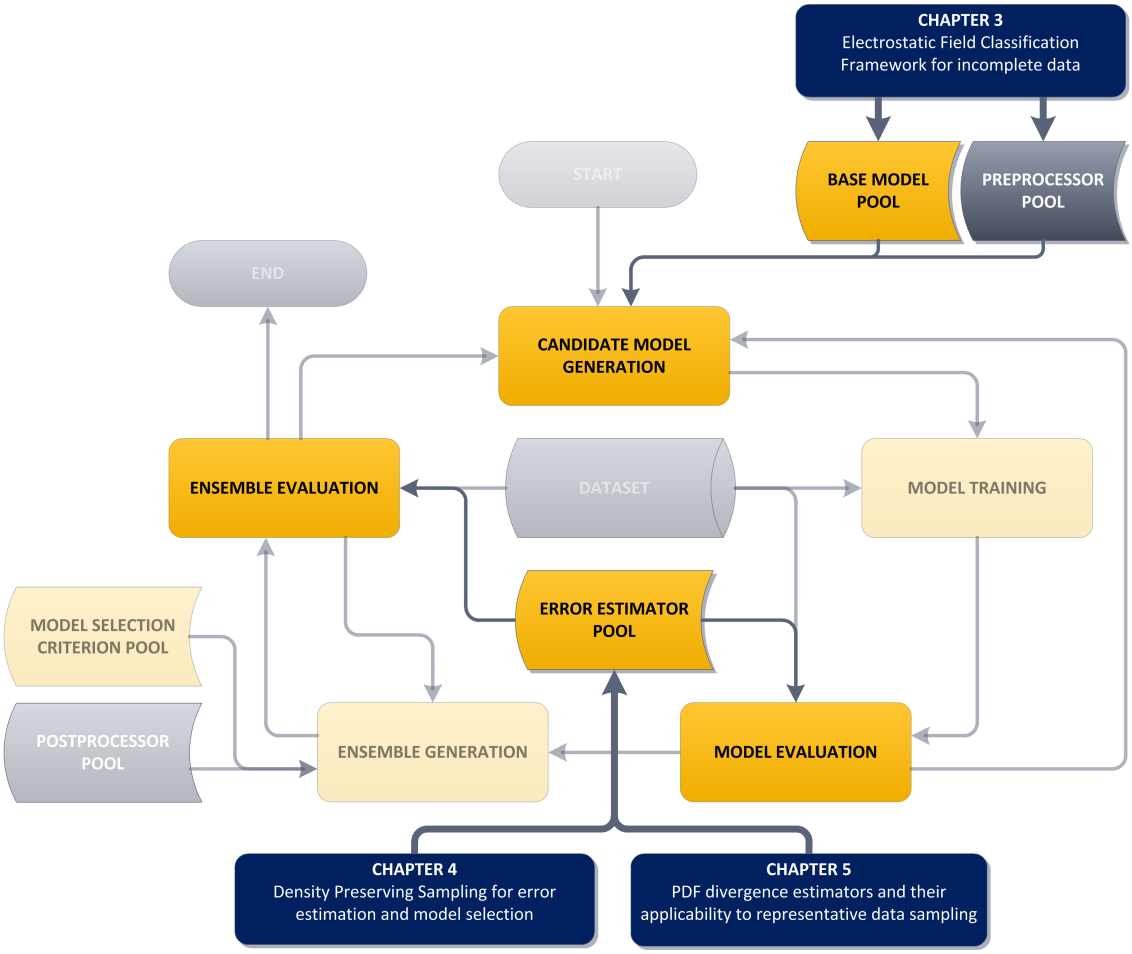


Figure 2.11: Correspondence between the generalised predictive system design cycle and physically inspired methods described in the thesis

Chapter 3

Electrostatic Field Classification Framework for incomplete data

3.1 Introduction

Supervised learning is the approach which enables design of robust and well–performing classifiers. Unfortunately, as already mentioned, in many real–world applications labelling of the data is costly and thus possible only to some extent. Unlabelled data on the other hand is often available in large quantities but a classifier built using unsupervised learning is likely to demonstrate performance inferior to its supervised counterpart. The interest in a so called semi–supervised learning is thus a natural consequence of this state of things and various approaches have been discussed in the literature [13, 32, 55, 62, 111, 114, 123].

While the missing labels can be thought of as one type of deficiency in the data on the basis of which one would like to build a well–performing classification system, another very commonly encountered problem is that of missing values of the input variables. There are many reasons why input data might be missing and there are many ways of dealing with it though the most commonly used approaches can be found in the statistics literature. The ideas and various types of missingness introduced in [136] are still in use today and the multiple imputation method [137] is considered as state of the art alongside the Expectation Maximisation (EM) algorithm [34, 58, 150, 171].

The semi–supervised learning and handling of missing input data are often treated as separate problems. A different approach to classification based on hyperbox fuzzy sets, not requiring imputation of missing values has been presented in [54]. The General Fuzzy Min–Max (GFMM) algorithms for clustering and classification naturally support incomplete datasets, exploiting all available information in order to reduce the number of viable alternatives before making the classification decision. The GFMM algorithm is also able to learn from both labelled and unlabelled data, processing both types of patterns for adaptation and labelling of the fuzzy hyperboxes simultaneously. Such philosophy of dealing with both unlabelled and missing input data within a consistent, unified framework is also pursued in this work.

The semi–supervised learning approaches and the methods of handling missing data discussed in Section 3.4 have various motivations, resulting from a number of probabilistic, fuzzy set and other machine learning theories. However, it is the nature inspired approach which appears as an attractive way to address the missing data problem, as the omnipresent uncertainty caused by missing data seems intrinsic to various natural systems. From different possible approaches, a physical inspiration has been chosen to design a missing data handling framework described in

this work, taking advantage of pioneering work on application of physical fields to classification problems presented in [147] and described in Section 2.3.1. Due to its properties, the Electrostatic Field Classifier has been chosen as a basis for building an unified missing data handling framework able to learn from both labelled and unlabelled, as well as complete and incomplete data using the same rules and adaptation mechanisms.

The framework described in this chapter is an attempt to directly exploit well known physical laws in the machine learning context. The analogy used, that is the data vectors acting as charged particles able to directly interact with each other through forces they induce, is one of the simplest and most basic ways to combine the two disciplines.

3.2 Data field classifiers

The classification framework for incomplete data derived in this chapter takes advantage of two classification algorithms based on physical field models. The algorithms have already been introduced in Section 2.3.1. As mentioned before, both of them treat each data instance as a charged particle able to directly interact with other particles. Following [147], a more detailed description is given below.

3.2.1 Gravity Field Classifier

Denoting by $\mathcal{X}$ a set of $N_{\mathcal{X}}$ training instances and by $\mathcal{Y}$ a set of $N_{\mathcal{Y}}$ test instances, the potential of the field generated by each instance in the j^{th} testing point is given by:

$$V_{ij} = -cs_i \frac{1}{r_{ij}} \tag{3.1}$$

where c is the field constant, s_i is the charge of instance $\boldsymbol{x}_i$ and r_{ij} is some distance measure between $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$. For simplicity and for the sake of conformity with the physical model Euclidean distance has been chosen. Note however, that the definition of the distance function can have tremendous influence on the field landscape and can also be used to incorporate support for categorical data. As shown in Sections 3.3.4 and 3.5.1, appropriate distance definition facilitates classification of incomplete instances and is able to overcome some problems in higher dimensional spaces ('curse of dimensionality').

As already stated, the superposition of individual contributions of all training instances defines the field in any point of the input space as:

$$V_j = -c \sum_{i=1}^{N_{\mathcal{X}}} \frac{s_i}{r_{ij}} \tag{3.2}$$

and the overall potential energy in point $\boldsymbol{y}_j$ is given by:

$$U_j = s_j V_j = -cs_j \sum_{i=1}^{N_{\mathcal{X}}} \frac{s_i}{r_{ij}} \tag{3.3}$$

where s_j stands for the charge of $\boldsymbol{y}_j$. Assuming that all instances are equally important (have the same, unit charge), s_i and s_j can be dropped from the equations and the force exerted on $\boldsymbol{y}_j$ by the field, being in fact the negative gradient of U_j takes the following form:

$$\boldsymbol{F}_j = -c \sum_{i=1}^{N_{\mathcal{X}}} \frac{\boldsymbol{y}_j - \boldsymbol{x}_i}{r_{ij}^3} \tag{3.4}$$

By analogy to the gravity field, the force $\boldsymbol{F}_j$ will try to drag the test instance into the lowest energy level. The field constant c is in fact only a scaling factor, which can also be dropped due to the fact that only the force direction is of interest in the model.

The classification procedure with the gravity field model follows a simulatory procedure and has already been described in Section 2.3.1. Due to the fact that the simulation step r is fixed in all dimensions, the data should be first rescaled to fit within the $[0, +1]$ range. The lower bound of the distance should also be set to some value comparable to r to avoid division by zero and overshooting the field sources.

3.2.2 Electrostatic Field Classifier

The Gravity Field Classifier does not fully exploit class label information given with the training data, effectively wasting information which could be used for improving accuracy. To address this issue the electrostatic field analogy has been employed resulting in a model with both attracting and repelling interactions, which as already stated in Section 2.3.1 required decomposition of each test instance into a number of subinstances, belonging to one of the target classes (partial memberships).

Denoting by $\boldsymbol{l}$ the vector of labels of the field sources and by V_j^k the potential generated by k^{th} of c classes in point $\boldsymbol{y}_j$, the partial membership p_{jk} of instance $\boldsymbol{y}_j$ in the k^{th} class is given by:

$$p_{jk} = \frac{\left|V_j^k\right|}{\sum_{i=1}^{c} \left|V_j^i\right|} \tag{3.5}$$

The collection of partial memberships for all test instances form a partition matrix $\boldsymbol{P}^{[N_{\mathcal{X}} \times c]}$. The overall potential in point $\boldsymbol{y}_j$ can be calculated as:

$$V_j = \sum_{i=1}^{N_{\mathcal{X}}} \left(\frac{\sum_{k \neq l_i} p_{jk} - p_{jl_i}}{r_{ij}} \right) = \sum_{i=1}^{N_{\mathcal{X}}} \frac{1 - 2p_{jl_i}}{r_{ij}} \tag{3.6}$$

The resultant force calculation formula then becomes:

$$\boldsymbol{F}_j = \sum_{i=1}^{N_{\mathcal{X}}} \left[(1 - 2p_{jl_i}) \frac{\boldsymbol{y}_j - \boldsymbol{x}_i}{r_{ij}^3} \right] \tag{3.7}$$

Note however, that if there are more than two classes, repelling force may dominate the field, as it would come from multiple classes, while the attracting force would come from only one. According to [147], to restore the balance between repelling and attracting forces it is sufficient to satisfy the following condition:

$$\sum_{j=1}^{N_{\mathcal{Y}}} V_j = \sum_{j=1}^{N_{\mathcal{Y}}} \sum_{i=1}^{N_{\mathcal{X}}} \frac{1 - qp_{jl_i}}{r_{ij}} = 0 \tag{3.8}$$

by estimating a value of the regularisation coefficient q. This coefficient controls the balance

between the total amount of attracting and repelling force in the field but as discussed in Section 3.3.1, the condition given above may in fact not be sufficient. This can result in some test instances being repelled by the field, which would prevent the algorithm from converging.

The classification process follows the same rules as in the case of GFC.

3.3 Improvements of the original Electrostatic Field Classifier

The Electrostatic Field Classifier described in Section 3.2.2 has a number of previously not addressed issues, causing problems when processing some datasets, especially in high–dimensional spaces. Before deriving the missing data handling framework, these issues are first discussed and appropriate solutions are proposed, which although heuristic, have proven to work well in the experimental setting.

3.3.1 Excess of repelling force

The excess of repelling force present in the field is one of the major problems of the original EFC. As stated in Section 3.2.2, a value of the regularisation coefficient q satisfying Eq. 3.8 may still be too small to restore the balance between attracting and repelling forces. As a result, during the simulation some of the test instances escape from the field. However, it has been observed during the experiments that as the simulation proceeds and the number of test instances remaining to be classified decreases, the regularisation coefficient (which is recalculated during each step of the simulation) tends to increase on average. This in turn makes the attracting interaction stronger and stronger and causes the repelled test instances to eventually return and be classified. There is a number of issues however:

- The classification accuracy is reduced – in the case of test datasets used in experiments described in Section 3.6, it is in fact worse than the accuracy of much simpler and faster Gravity Field Classifier.
- The form of Eq. 3.8 makes the field landscape dependant on the test set. In the case of two different but partly overlapping test sets this may lead to different classification decisions regarding the overlapping part, depending only on the choice of the test set.
- The convergence is needlessly slow as the repelled instances usually travel very far from the field sources and it takes a large number of simulation steps until the field drags them back.
- Recalculation of the regularisation coefficient is computationally expensive and repeating it during each step of the simulation negatively affects classification speed.

To address the above issues a new procedure for estimating the regularisation coefficient has been devised. This estimation procedure is performed before the classification process and can thus be seen as a training phase of the classifier.

The procedure starts with calculation of the field boundaries, taking the minimal and maximal values of all features of the training dataset (0 and 1 after scaling) and adjusting them by a small value equal to the minimal step size (see Section 3.3.2). Then an artificial test set is generated by placing instances in the corners of the field defined by the boundaries. However, as the data dimensionality grows, this approach can quickly become prohibitive – the number N of corner points of $d-$dimensional cube is an exponential function of data dimensionality – $N = 2^d$. This issue has been addressed by using only a small random subset of such artificial test set, which in

most experiments did not degrade the classification performance. If data dimensionality allows doing so, a number of test instances can also be placed along the field boundaries, which has been found to improve the classification performance especially in two–dimensional spaces. After the artificial test set has been generated, the regularisation coefficient is estimated by forcing all field vectors located on the test instances to point into the field, as shown in Figure 3.1.

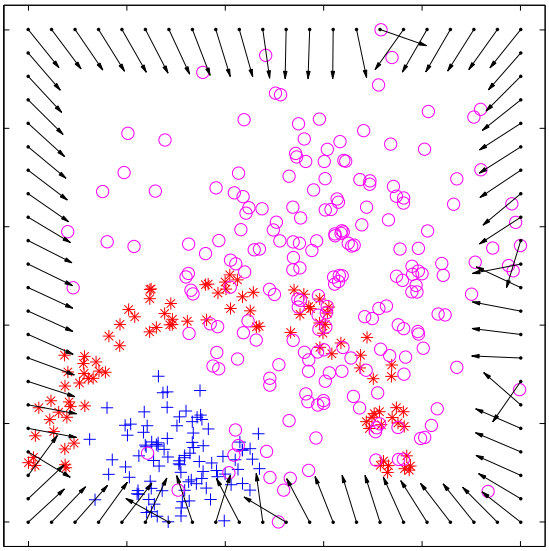


Figure 3.1: Regularisation coefficient estimation

3.3.2 Simulation step size

The simulation step size is another crucial parameter of EFC, as it largely determines the convergence speed and the stability of the algorithm. If the step size is too small, the convergence will be needlessly slow and the model will be sensitive to noisy field sources; if it's too big, the test instances will overshoot the sources and start oscillating preventing convergence. In the previous work however [147], the issue of optimal step size has not been addressed.

The step size should somehow depend on the training data and it does not need to be held constant during classification. As a result the following improvements have been introduced in the current implementation:

- each test instance has been assigned its own step size r_j,
- the step sizes are constantly recalculated during the simulation according to:

$$r_j = \max\left(r_{min}/d,\ \min(r_{ij})/d\right) \tag{3.9}$$

  where $r_{min} = \min(r_{ii})$ is the minimal distance between field sources, and d is the number of dimensions.

The above modifications enable the step size to adapt to the current situation taking advantage of the fact that the field far away from the sources does not change rapidly. As a result the bigger the distance between a test instance and the nearest field source, the bigger the step.

3.3.3 Early label assignment and label conflict resolution

The classification process described in Section 3.2.2 runs until a test instance approaches one of the field sources at the distance smaller or equal to r_{min}. The test instance then shares the label of this source and hence the classification is accomplished. There are however some situations in which this behaviour is not desired – an example is shown in Figure 3.2. As it can be seen, forces acting on all test instances (marked with arrows) will drag them towards the cluster in the top left corner of the field. On their way some instances will however approach a single source, which is an outlier belonging to another class. Part of them may even end up trapped in its field, while other may simply pass it by. For the original EFC this would not make any difference – all those instances would end up being classified as belonging to the class of the outlier (represented with circles). Although it is justified for the test instances trapped by this single field source, other instances would be labelled too early. For this reason the label assignment criterion has been changed – the test instance must not only approach a source but also stabilise or start oscillating around it. This oscillation is detected by checking if the total displacement of a test instance in two consecutive simulation steps exceeds r_{min}. If it does, the test instance is considered not stable and the classification decision is postponed. Although this new convergence criterion does not influence the classification performance greatly when dealing with complete datasets, the situation changes dramatically when incomplete data comes into play. The reason for this is mainly the representation of deficient test instances in the model, as described in Section 3.5.1.

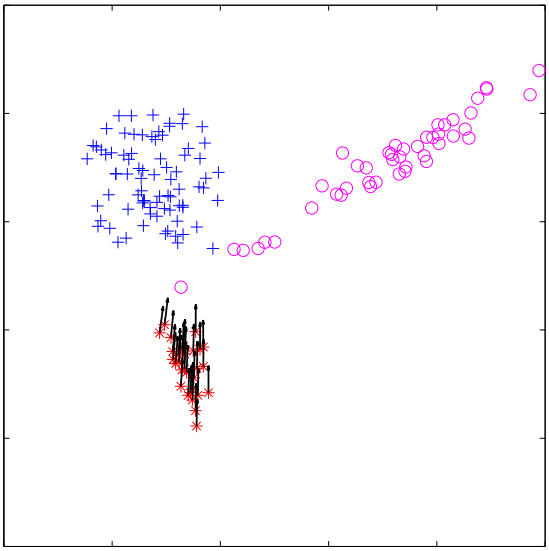


Figure 3.2: Early label assignment issue

There are also situations when a test instance stabilises around a number of field sources, all coming from different classes. This is often the case when dealing with incomplete data, as will be discussed later. Some conflict resolution mechanism must then be applied to arrive at the classification decision. In such case EFC falls back to GFC and uses the GFC potential (which is almost readily available in the form of the partition matrix) to select the winning class. Note, that the partition matrix concept can also be used for producing soft outputs by returning a whole row from this matrix instead of a crisp class label.

3.3.4 Curse of dimensionality and distance concentration

For many tested datasets both GFC and EFC tend to fail when the dimensionality of the data grows roughly above 5 or 6, depending on the dataset, which leads to either (1) very high classification error when compared to the performance of other standard classifiers or (2) no convergence at all.

The latter issue has been investigated using 13–dimensional Wine dataset (for details on the datasets see Appendix A). The experiments revealed that both algorithms fail to converge due to a part of the test instances concentrating in a small number of regions (typically 2–3 for the Wine dataset) somewhere within the field boundaries but not close enough to any of the field sources to be classified. Moreover, although it is possible to classify the trapped test instances using the conflict resolution mechanism described in Section 3.3.3 (pick the label of the class with highest potential), this approach produces very high classification error when compared to other classifiers.

This phenomenon can be explained by the choice of distance measure, namely the Euclidean distance, which is based on the L_2-norm. The L_2-norm is a member of the family of the L_p-norms, which are defined for any $d-$dimensional vector $\boldsymbol{x}$ as:

$$L_p(\boldsymbol{x}) \ = \left(\sum_{i=1}^{d} |x_i|^p \right)^{\frac{1}{p}} \tag{3.10}$$

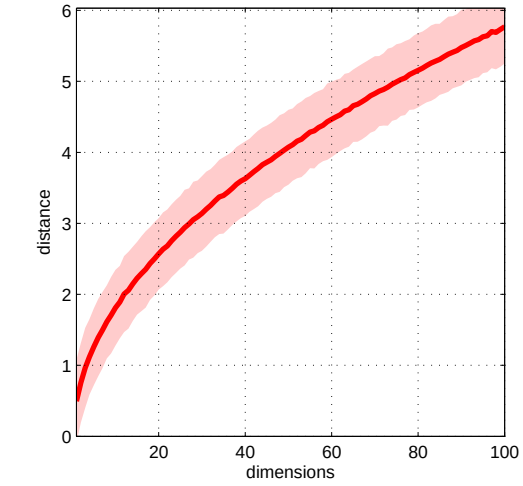
assuming that $p \geq 1$. A $p-$norm distance between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is thus given by:

$$D_p(\boldsymbol{x}, \boldsymbol{y}) \ = L_p(\boldsymbol{x} - \boldsymbol{y}) = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{\frac{1}{p}} \tag{3.11}$$

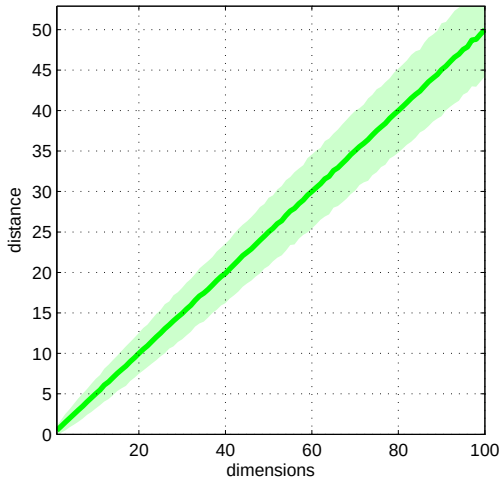
The Euclidean distance is an intuitive and naturally interpretative choice for 2 or 3 dimensional spaces. In the case of the physical field models, it is additionally justified by the fact that 'real' physical fields also exist in 3 dimensions only. However, as the number of dimensions grows, the Euclidean distance looses its discriminative power, regardless of the characteristics of the dataset, which makes its natural interpretability irrelevant [2, 50]. The same applies to other distances based on L_p-norms, but fortunately to a different degree. The reason for this is that under a broad set of conditions the mean value of the L_2-norm distribution grows with data dimensionality while the variance remains approximately constant (Figure 3.3(a)) [50]. As a result, the nearest and furthest neighbours of any test point appear to be at approximately the same distance, which makes the ratio of distances to the nearest and farthest neighbour tend to 1. The phenomenon has been called 'distance concentration' [2, 50]. As argued in [10], it can occur even for sets with as few as 10 dimensions and the decrease in the ratio between the farthest and nearest neighbour distance is steepest in the first 20 dimensions. The effect is additionally magnified by the limited precision of calculations a computer can handle.

According to [2] with high probability growing with the number of dimensions, the concentration is slower for norms of lower order. It has been proven that the L_1-norm based distance (the Manhattan/block distance) is more suitable for high–dimensional data than the L_2-norm based distance [2].

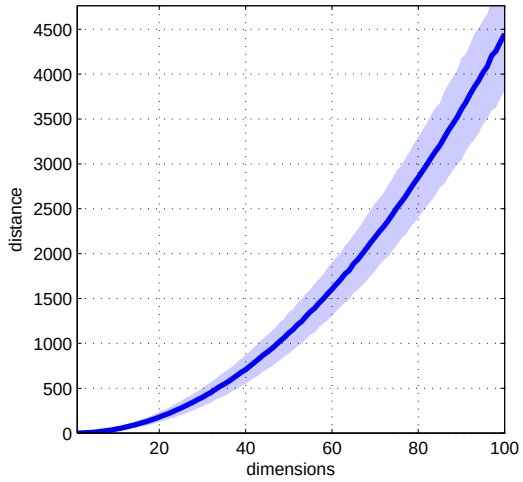
If $p < 1$ the resulting measure wouldn't be a true distance anymore due to violation of the triangle inequality. It might still however be a useful similarity measure and a family of this kind of measures has been developed in [2] and called 'fractional distances'. Although for some specific conditions a fractional distance can concentrate even faster than distances of higher



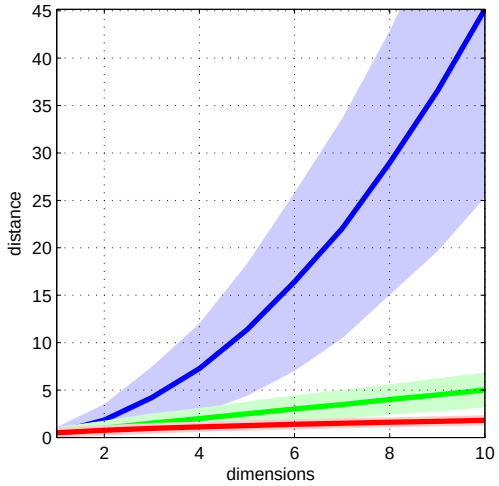
(a) Euclidean distance



(b) Manhattan distance



(c) Fractional distance (0.5)



(d) All 3 distances

Figure 3.3: Distance concentration for various measures, for a random vector drawn from a unit hypercube (solid line denotes the mean value, shaded region denotes the mean $+/-$ 2 standard deviations). Notice the units on the left. All three distances have been compared on the rightmost plot (Euclidean – bottom line, Manhattan – middle line, fractional of order 0.5 – topmost line).

orders, in most situations it is much less concentration prone [50]. The concentration issue for various norms has been depicted in Figure 3.3. Note, that by using a fractional distance the feature space gets deformed and thus the distance measure looses its intuitive interpretability. This can be seen in Figure 3.4, where the unit 'circles' for three distances of different orders have been presented.

There have been other attempts to address the concentration issue, one of which involved calculation of the similarity measure using only some subset of features meaningful for the pair of data instances in question [1]. The features to be used can be selected according to various

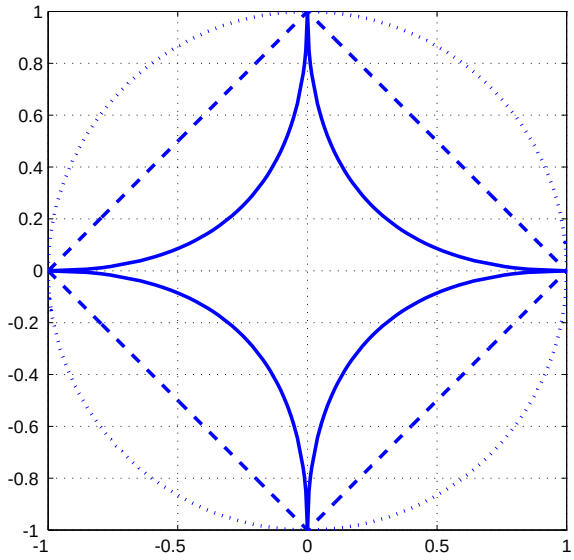


Figure 3.4: Unit circles for L_2-norm (dotted), L_1-norm (dashed) and $L\frac{1}{2}-$norm (solid) based distance

criteria (e.g. exceeding some threshold value). This however implies that the subset of features can be different for any pair of patterns, which makes such similarity measure non differentiable, thus requiring computationally expensive numerical estimation of the gradients.

As a result all experiments were run using distance measure of order derived experimentally, separately for each of the datasets, by exhaustive search in the range between 0.01 and 2. Beside achieving good classification performance as shown in Section 3.6, this approach also allowed to eliminate the concentration issue described in this section almost completely – it now occurs occasionally and in most cases applies only to a single, usually incomplete test pattern and is efficiently addressed using the conflict resolution mechanism described in Section 3.3.3.

3.4 The missing data problem

The missing data problem is typical for many research areas and can be divided into two subproblems: (1) learning from incomplete data and (2) making decisions given incomplete input. Note, that the 'incompleteness' can denote missing attributes (inputs) as well as missing labels or targets (outputs). There are two main types of procedures for dealing with missing data [149]:

- Imputation, that is recovering or otherwise estimating the missing data, which allows the subsequent analysis to be run as if the data was complete.
- Estimation of the parameters of underlying models (learning) and making decisions directly from the incomplete data, without imputation. This is the approach adapted in the framework derived in Section 3.5.

3.4.1 Types of missingness

There are many reasons why the data may be missing, depending on the nature of the data, the data collection process and other factors. The missingness can be caused by equipment malfunction, if the data is collected in an automated manner; the data can be entered incorrectly, someone may

refuse to answer a particular question in a survey and so on. The reason why the data is missing can have important implications on the choice of method to deal with the missingness. For this purpose, according to [136], missing data can be divided into three categories:

- **Missing Completely At Random (MCAR)**, when the probability that a particular feature is missing is not related to the missing value or to the values of other features. For example, if a temperature sensor fails just because they do it sometimes the data is MCAR. If the same sensor fails because the temperature was so high that it got damaged – the probability of missingness is related to the value of missing measurement and thus the data cannot be considered MCAR [117, 150].
- **Missing At Random (MAR)**, if the probability that the particular feature is missing is not related to its own value but is related to the values of other features. It is possible to deal with MAR data to obtain meaningful and relatively unbiased estimates [117] but unfortunately there is no way to test if MAR holds in a given dataset. In many practical cases however a false assumption of MAR may have only minor influence on the result [150].
- **Missing Not At Random (MNAR)**, if the probability that some feature is missing is a function of this feature value. The mechanism for missingness is not ignorable and should be somehow modelled. Unfortunately, the model for missingness is rarely known, which makes the whole procedure a difficult and application specific task [117].

Since both GFC and EFC are purely data–driven approaches, which means that no domain or any other external knowledge is incorporated into the models, the type of missingness does not directly influence their operation. Although it seems reasonable to assume that the performance of the models is dependant on the type of missingness, this dependency has not been investigated here and all the experiments have been designed in a way enforcing the MCAR assumption.

3.4.2 Traditional approaches to missingness

There is a number of basic, traditional approaches to the missing data problem. In the case of missing features most of those methods are based on editing or imputation of missing values, which makes them easily integrable with a wide range of machine learning and statistical models. Many imputation techniques have however received a bad press and although usefulness of most of them is limited to a number of specific cases, they still are in common use due to the ease of integration with standard models [117, 150].

Missing features

As mentioned before, the methods for dealing with missing features can be divided into two groups: (1) editing and imputation methods, and (2) direct methods. Some of the best known methods belonging the the first group are:

- **Casewise deletion (CWD)**, which assumes that incomplete data instances can be ignored and excluded from further analysis. If their number is relatively small, this technique turns out to perform surprisingly well – if the data is at least MAR, casewise deletion produces unbiased estimates of the parameters [63].
- **Mean imputation (MIMP)**, which replaces all missing features with appropriate mean values. Although it may seem reasonable, the method does not add any new information,

as the overall mean will remain the same. In statistical analysis this can lead to underestimation of the standard deviation – the substituted values do not contribute to the value of deviation but they do contribute to the number of instances used for the calculation (denominator of the standard deviation formula). In machine learning mean imputation can distort the distribution of the data, which is especially dangerous if the distribution departs from Gaussian.

- **Class–conditional mean imputation**, which replaces the missing attributes with the mean values calculated for the respective classes [151]. The method can thus be used for labelled data only and is a preferred choice over the standard mean imputation. For this reason, in the subsequent parts of this work only class–conditional mean imputation is used.
- **Imputation from a distribution**, which is a family of techniques guided by local dataset properties, replacing a missing feature with the most likely value taking into account the non–missing features. These approaches usually require estimation of a joint probability density function and integration over all missing dimensions, which is a problem exponential in the number of missing features [107]. A simple variant of this method called 'hot–deck' imputation involves replacing missing features with the corresponding values from a randomly picked, complete pattern from the same dataset.
- **Multiple imputation**, which is an extension of imputation from a distribution. The method generates multiple versions of the dataset, analyses each of them using a complete data method and combines the results by averaging them. In statistical inference, this not only allows to obtain the overall estimates but also their standard deviations, which reflect uncertainty caused by the missing data [137].

Examples of the direct missing data handling approaches are:

- **Pairwise deletion**, which ignores missing data but as opposed to casewise deletion, rather than dropping incomplete instances, takes advantage only of the features which are present. For example, in the case of a dataset of N two–dimensional instances, where half of the instances have the second feature missing, the mean of the first feature will be calculated using N data points, while the mean of the second feature will be calculated using only $N/2$ data points. Although all available data is exploited, various parameters are computed using datasets with different sizes and different standard errors, which can lead to unpredictable results (e.g correlation coefficient falling outside of the $[-1, +1]$ interval) [150]. For those reasons this method is not recommended for estimation of statistics. However, a similar approach forms a basis of the proposed data field specific approach to classification of incomplete patterns, with a great success as described in Sections 3.5.1 and 3.6.
- **Maximum likelihood (ML) methods**, which are approaches for estimation of parametric models. Although those methods require integration over the missing dimensions to obtain the likelihood function, there is a feasible, iterative way for calculation of maximum likelihood estimates – the Expectation Maximisation algorithm [34], which has greatly contributed to the success of maximum likelihood estimation methods.

Missing labels

Learning from a dataset with missing labels is known as the semi–supervised learning problem. The supervised approach usually achieves better classification performance, but in many applications labelling the data can be expensive and time consuming. On the other hand, large

amounts of unlabelled data are often readily available, but unsupervised learning will seldom produce a classifier with comparable performance [55]. The semi–supervised approach thus aims and exploiting both these types of data at the same time.

There are some standard methods of dealing with datasets with incomplete label information. A following classification has been introduced in [55]:

- **Pre–labelling**. This approach first builds an initial classifier using the labelled part of the dataset only. The unlabeled data is then classified using this initial model and a final classifier is designed using the whole, now completely labelled dataset [13, 32, 62, 111, 114]. This method is also referred to as **Pre–classification (PC)** in the remainder of this work and is used in the experiments as a baseline for performance comparison.
- **Post–labelling**. In this method the whole dataset is first clustered and then the labelled data is used for labelling the clusters using the majority rule [94]. Alternatively a data density estimator can be used and the labels can be then assigned in accordance to the highest class conditional density in the location of the unlabelled instance.
- **Semi–supervised**. An approach which assumes processing of both labelled and unlabelled data at the same time. This method of learning can take a form of semi–supervised clustering guided by the class labels, in which the data is iteratively split until the majority of instances in each of the clusters belongs to the same class [123]. The GFC–fallback method derived as a part of the proposed missing data handling framework and described in Section 3.5.2 is an example of a semi–supervised approach.

In the experiments described in Section 3.6 the labels have been removed randomly. Although the results are strongly dependant on the representativeness of the labelled subset of data, this problem is a part of the active learning domain [135] and is not investigated here.

3.5 Electrostatic field framework for incomplete data

The overview of the architecture of the proposed framework has been given in Figure 3.5. The central part of the framework (the ‘engine’) is the EFC/GFC model pair, with EFC acting as the main classifier and GFC playing the role of a helper solution (estimation of the partition matrix, handling of unlabelled data). As described in Section 2.3.1, both models work in a simulatory manner, minimising the overall energy of the field. Note the modular design of the framework, which allows to easily plug–in various definitions of distance and force as well as different label assignment rules and preprocessing routines. In order to introduce the incomplete data support it is sufficient to modify these external components without touching the framework engine at all, which includes:

1. **Distance definition** to facilitate calculating distances not only between pairs of complete patterns but also between a complete and incomplete pattern.
2. **Force definition** to facilitate calculating forces exerted by field sources not only on complete test patterns but also on incomplete test patterns.
3. **Label assignment procedure** to facilitate assigning labels not only to complete test patterns but also to incomplete test patterns.
4. **Preprocessing routine** to extract information from incomplete training data.

Descriptions of these modifications in all possible incomplete data scenarios are given in the next sections. The only requirement imposed by the framework is that each pattern has at least one feature value specified.

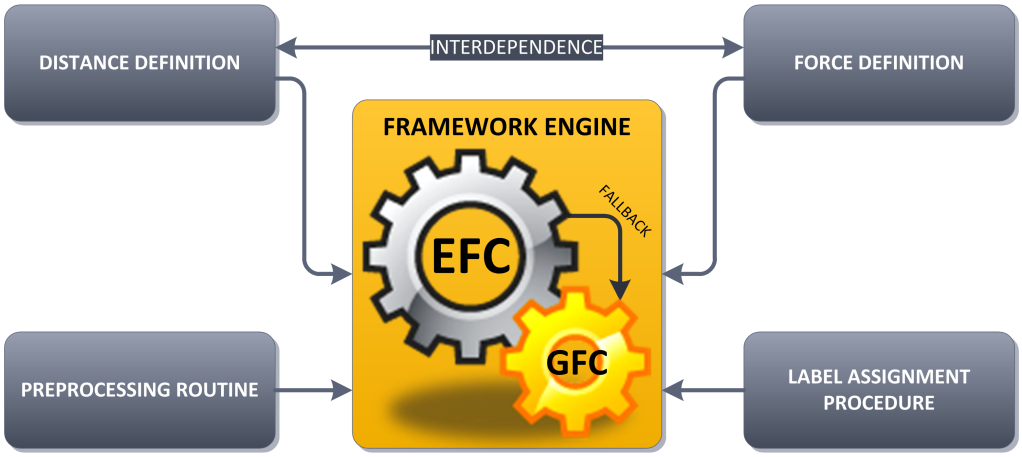
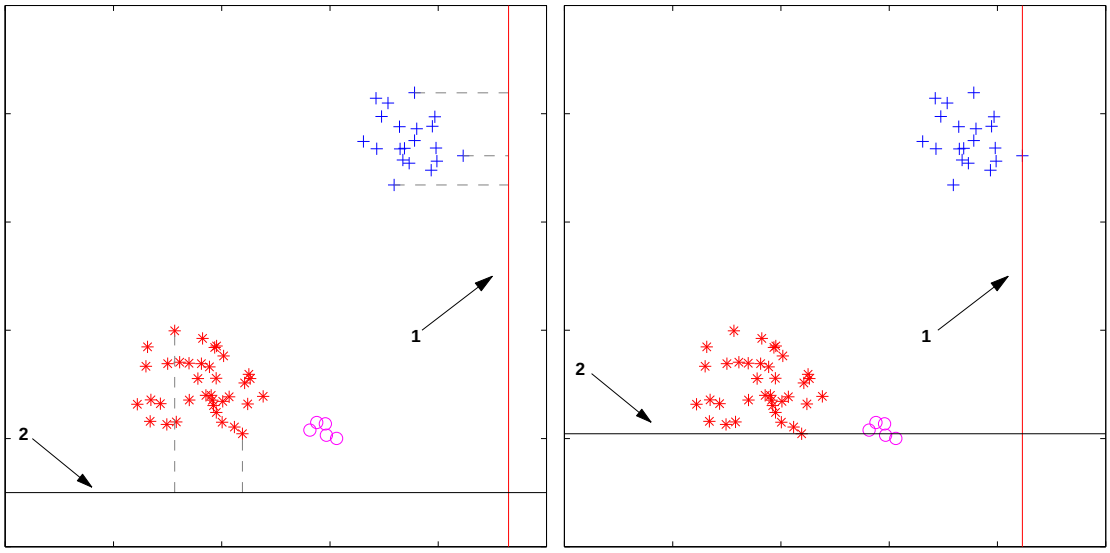


Figure 3.5: Overview of the framework architecture

3.5.1 Incomplete test data

The proposed approach to classification of incomplete data instances is to use all available information to arrive at a classification decision rather than trying to estimate the values of missing features and then carrying on with the classification procedure. In this framework, the EFC acts on incomplete test data working only in available dimensions – the feature space is automatically, locally reduced to accommodate the number of available attributes. This is somewhat similar to the pairwise deletion described in Section 3.4.2. The modifications to the original EFC include:

1. **Distance definition.** The distances are now calculated using available dimensions only. As a result in the example given in Figure 3.6(a) the distance between test instances and any of the field sources is equal to the distance between the source and the line (plane or hyperplane) representing the test instance. Some examples of distances have been marked with dashed lines. From now on this distance will be referred to as r^*.
2. **Force calculation procedure.** The forces can only be exerted in the available dimensions, hence instance 1 in Figure 3.6 can only be dragged to left or right and instance 2 – up or down.
3. **Label assignment procedure.** When the incomplete test instance stabilises or starts oscillating in the neighbourhood of any source the classification decision can be made. Unfortunately an incomplete pattern cannot always simply share the label of the nearest field source. As it can be seen in Figure 3.6(b), in the case of instance 2 this criterion is not sufficient to make a classification decision because of the apparent ambiguity. The approach used is to produce a soft, probability–like output being in fact proportional to the class conditional density for the current position of the test instance, calculated using only existing feature values. This information is almost readily available in the form of the partition matrix (see Eq. 3.5).



(a) Representation (b) Classification

Figure 3.6: Representation and classification of incomplete test patterns

3.5.2 Incomplete training data

The situation with incomplete training dataset is different from what has been said in Section 3.5.1. First of all, in the case of training data the missingness can apply not only to the features but to the labels as well. Moreover, for the reasons described below, the approach taken to the problem of incomplete test dataset is not suitable in this situation.

Missing features

Attempt to exploit patterns with missing features as field sources during the classification process using only the features which are available did not succeed. Even a single incomplete field source with even a single feature missing changes the field landscape so much that the recognition rate falls far below random guessing. The reason is the representation of the incomplete source by an infinite number of points covering the whole missing feature space, which can be pictured as the whole line, plane or hyperplane acting as a source. For this reason incomplete training patterns do not take part in the classification process but are exploited during the preprocessing phase instead.

In the original GFC and EFC models it has been assumed for simplicity that all data instances carry the same unit charges. However, by differentiating the charges of the field sources and employing an intelligent Charge Redistribution (CR) mechanism, the information contained in incomplete training patterns can be at least partially exploited.

The Charge Redistribution algorithm starts with assigning a unit charge to all training instances (both complete and incomplete). It then examines each incomplete instance in turn and redistributes its charge among all complete patterns from the same class in proportion to the distance from the incomplete instance in question, according to:

$$\Delta \boldsymbol{C}_{ci}^{comp} = \frac{s_i}{\boldsymbol{R}_{ci}^{comp*}} \Big/ \sum_j \frac{s_i}{r_{cij}^{comp*}} \tag{3.12}$$

where $\Delta \boldsymbol{C}_{ci}^{comp}$ denotes the vector of charge gains for all complete field sources coming from c^{th} class due to redistribution of charge of the i^{th} incomplete field source from the same class, s_i denotes the charge of the i^{th} incomplete field source (equal to unity for simplicity) and $\boldsymbol{R}_{ci}^{comp*}$ with elements r_{cij}^{comp*} denotes a vector of distances calculated using available features, between all complete field sources coming from c^{th} class and the i^{th} incomplete source from c^{th} class.

The proportion of charge assigned to any complete instance is in effect calculated using the GFC potential formula (Eq. 3.1), thus the closer the complete training instance to the incomplete instance in question, the more charge it receives. After all incomplete patterns are processed they are dropped and the remaining instances become field sources. This approach thus requires each class to have at least one complete instance, which will take over the charge of the incomplete objects.

There is also another possibility of redistribution of the incomplete field sources charge, called Multidimensional Charge Redistribution. The idea involves introduction of multidimensional charge concept, which means that the amount of charge in each of the dimensions can differ and that each incomplete training instance redistributes its charge only in the dimensions for which its features values are known. Preliminary experiments have however revealed that Multidimensional Charge Redistribution never performs better than the ordinary Charge Redistribution (in most cases it has in fact performed worse) and at the same time it introduces additional computational overhead. As a result, this method has been abandoned and all further experiments have been conducted using the Charge Redistribution formula of Eq. 3.12.

Figures 3.7 and 3.8 depict evolution of the decision boundaries for a 2–dimensional PCA projection of the Iris dataset (Appendix A) for EFC with Charge Redistribution and casewise deletion, as the deficiency level increases. Note, that for CR, the shapes of the decision boundaries resemble the original boundaries up to 80% deficiency level, which is not the case for casewise deletion, although at 60% deficiency it still allows to reconstruct the decision boundaries quite well. This should however be credited to the random nature of this experiment, which also applies to the maximum deficiency scenario and the apparent advantage of CR.

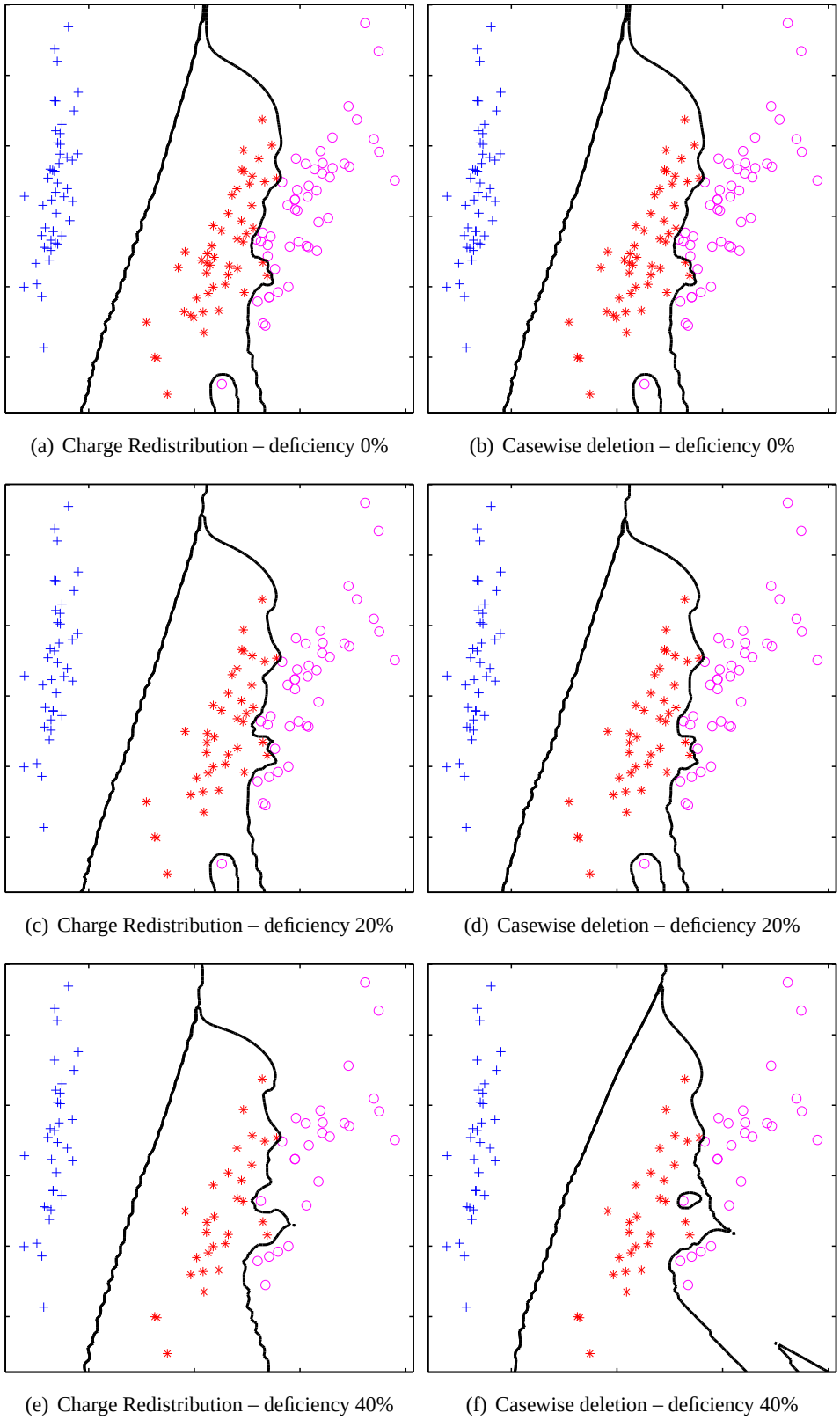
Missing labels

Three different approaches have been proposed for the EFC model to address the missing label issue. An important thing to notice is that none of them requires implementation of any additional mechanisms as all of them are already in place. The approaches are:

- **GFC–fallback**, which involves treating unlabelled instance as a gravity field source able to attract other instances but not able to repel and not able to share its label.
- **Charge Redistribution** similar to the mechanism described in Section 3.5.2 with the only difference being redistribution of charge among all complete field sources regardless of class.
- **Pre–classification** of unlabelled instances in order to assign them to existing classes.

3.5.3 Incomplete both test and training data

No additional modifications or adjustments are required when both test and training datasets are deficient. Modifications described in Sections 3.5.1 and 3.5.2 are fully compatible with each other. This means that the classifier can handle both types of missing data problems either separately or simultaneously, including the missing label case.



(a) Charge Redistribution – deficiency 0%

(b) Casewise deletion – deficiency 0%

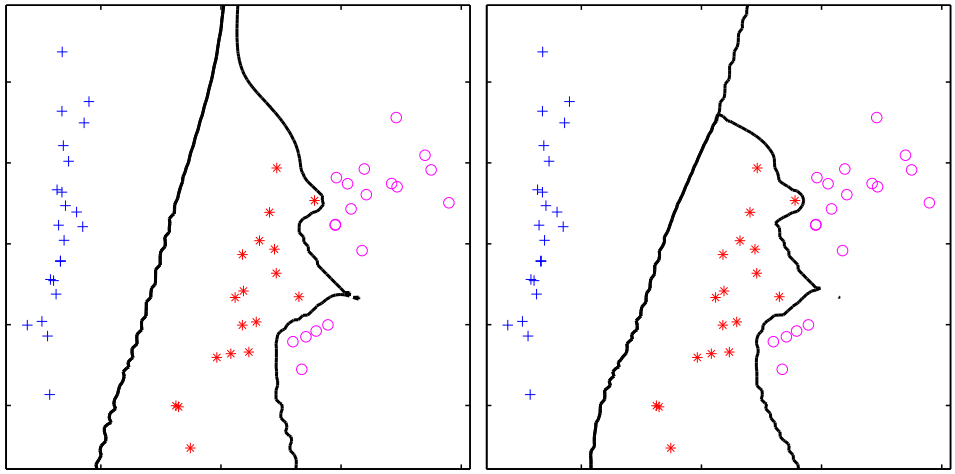
(c) Charge Redistribution – deficiency 20%

(d) Casewise deletion – deficiency 20%

(e) Charge Redistribution – deficiency 40%

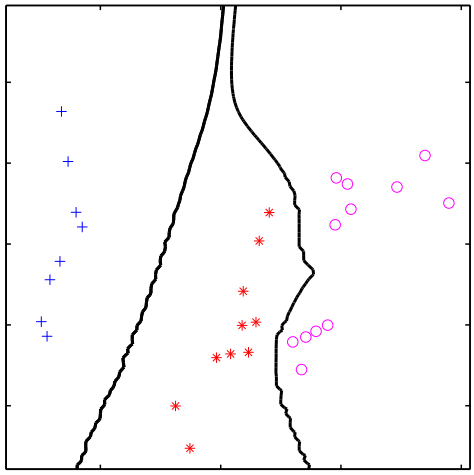
(f) Casewise deletion – deficiency 40%

Figure 3.7: EFC decision boundaries for a 2–D PCA projection of the Iris dataset for Charge Redistribution (left column) and casewise deletion (right column) as the deficiency level increases.

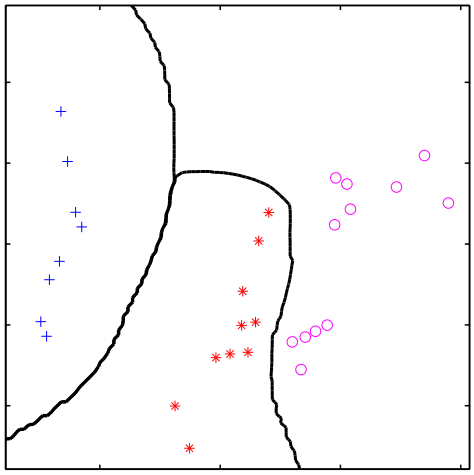


(a) Charge Redistribution – deficiency 60%

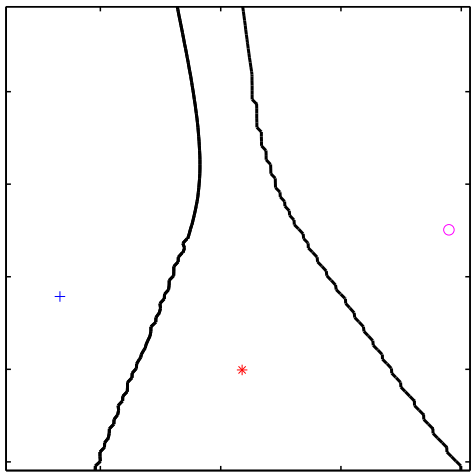
(b) Casewise deletion – deficiency 60%



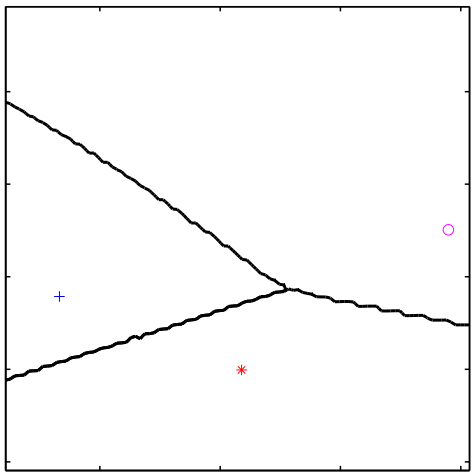
(c) Charge Redistribution – deficiency 80%



(d) Casewise deletion – deficiency 80%



(e) Charge Redistribution – deficiency 100%



(f) Casewise deletion – deficiency 100%

Figure 3.8: EFC decision boundaries for a 2–D PCA projection of the Iris dataset for Charge Redistribution (left column) and casewise deletion (right column) as the deficiency level increases (contd.)

3.6 Experiments

The experiments include evaluation of classification error for the scenarios of incomplete test data, incomplete training data, missing labels and various combinations of the above. All recognition rates given have been averaged over 10 runs with randomly removed features and labels. Also, each 10–fold cross–validation run has been repeated 10 times, which totals to 100 simulations for each deficiency level for every dataset. The deficiency step has been set to 0.15.

Due to the requirements of the framework, as well as other methods used for comparison, at the maximum deficiency levels[1] all classes have been left with exactly one training instance with all feature values given and exactly two labelled training instances. Also each of the remaining instances has been left with exactly one non–missing feature value.

A total of 12 benchmark datasets described in more detail in Appendix A has been used in the experiments. A short summary of the datasets used has been given in Table 3.1, in which the reduction of the number of simulation steps due to the adaptive adjustment of the step size r_j (Section 3.3.2) for a subset of the datasets has been given in the last column.

Table 3.1: Dataset details

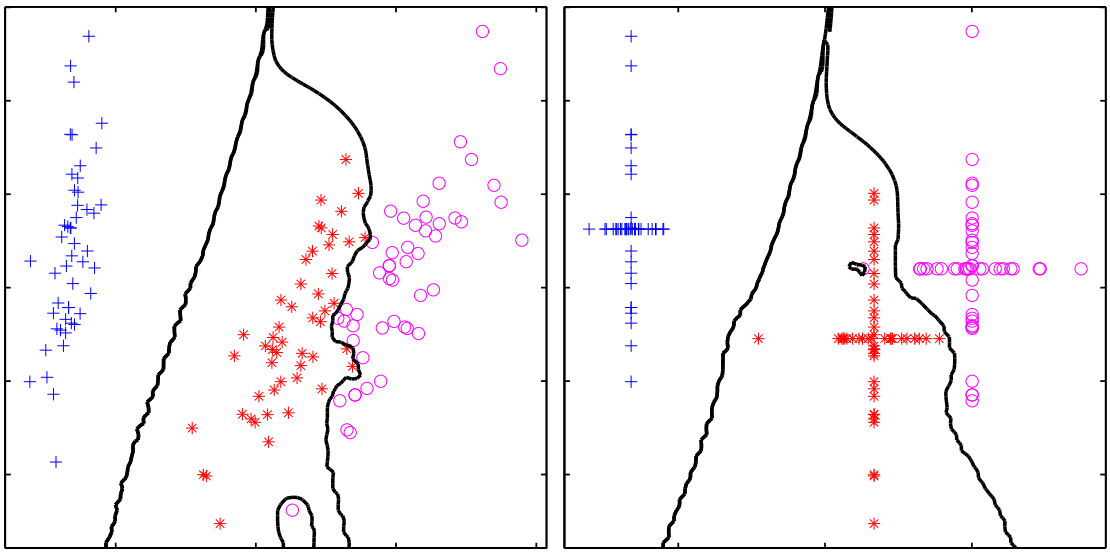
no.	abbr.	name	# objects	# attributes	# classes	steps
1	azi	Azizah dataset	291	8	20	–
2	bio	Biomedical diagnosis	194	5	2	–
3	can	Breast cancer Wisconsin	569	30	2	–
4	cnt	Cone–torus	400	2	3	43→23
5	gla	Glass identification data	214	10	6	–
6	ion	Ionosphere radar data	351	34	2	–
7	iri	IRIS flower database	150	4	3	21→16
8	liv	Liver disorder	345	6	2	–
9	syn	Synth–mat	250	2	2	20→20
10	thy	Thyroid gland data	215	5	3	–
11	veh	Vehicle silhouettes	846	18	4	–
12	win	Wine recognition data	178	13	3	32→15

The first value in the last column denotes the number of simulation steps needed for the algorithm to converge using optimal step size, found by exhaustive search. The second value is the number of steps using the adaptive step size. The classification performance for the adaptive approach was always very close to the standard method – the difference never exceeded 1 percentage point. As it can be seen, the reduction in the number of steps can vary greatly between 0 and 50%. Note however, that the total simulation time does not depend linearly on the number of simulation steps. Since the number of instances remaining to be classified drops as the simulation progresses, the amount of calculations each consecutive simulation step requires is constantly reduced as well.

The classifiers used for the comparison with the methods derived in this work have been listed in Appendix B.1. The classifiers which are available as a part of PRTools Toolbox [42], have been used with their default parameter values and each of them has been evaluated

[1]Deficiency level is the level of missingness of a dataset, with 0 (or 0%) for complete data and 1 (or 100%) for maximally incomplete data, taking into account the constraints given.

using class–conditional mean imputation (Section 3.4.2) to handle missing attributes and Pre–classification (Section 3.5.2) to deal with missing labels.



(a) Complete dataset (b) Maximally deficient dataset with imputed class means

Figure 3.9: Decision boundaries for complete and maximally deficient PCA projection of the Iris dataset

The performance of all models for incomplete test data, calculated as a sum of error rates for each class weighted by class prior probabilities estimated from the dataset, is given in Table 3.2. The results were averaged over all datasets and are given as error/standard deviation pairs, where the standard deviation has been calculated for 10 runs with randomly removed features and/or labels, simulating the MCAR scenario. Figure 3.10 presents the average recognition rates ($1 - error$) for both EFC algorithms used (*efc–ldr* and *efc–mimp*) and the median of errors of all classifiers.

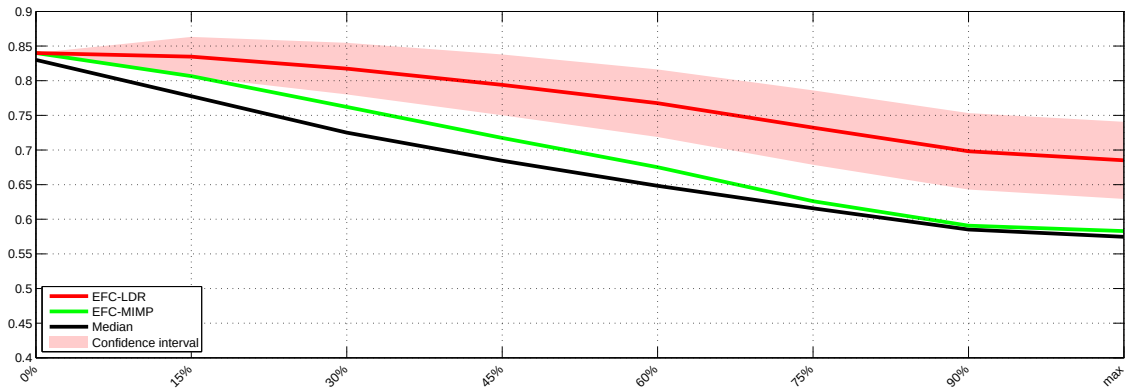


Figure 3.10: Performance for incomplete test data and various deficiency levels (recognition rates)

As it can be seen, for the incomplete test data scenario, the performance of *efc–ldr* model is superior to all other classifiers for all deficiency levels. Moreover, although at maximum deficiency the probability of error of the EFC model is twice as high as for complete data, this ratio reaches 3 for the classifiers which performed best at 0% deficiency. Also note, that in the complete

Table 3.2: Performance for incomplete test data and various deficiency levels

classifier	**deficiency level**							
	0%	**15%**	**30%**	**45%**	**60%**	**75%**	**90%**	**max**
efc–ldr	0.16	**0.17/0.03**	**0.18/0.04**	**0.21/0.04**	**0.23/0.05**	**0.27/0.05**	**0.30/0.06**	**0.31/0.06**
efc–mimp	0.16	0.19/0.04	0.24/0.04	0.28/0.05	0.32/0.05	0.37/0.05	0.41/0.04	0.42/0.04
fisherc	0.18	0.25/0.04	0.30/0.05	0.33/0.05	0.37/0.05	0.40/0.05	0.42/0.05	0.43/0.05
ldc	0.15	0.22/0.04	0.27/0.05	0.32/0.05	0.35/0.05	0.38/0.05	0.41/0.05	0.41/0.04
loglc	0.15	0.23/0.04	0.29/0.05	0.33/0.06	0.37/0.06	0.41/0.06	0.45/0.05	0.46/0.05
nmc	0.23	0.24/0.03	0.27/0.04	0.29/0.05	0.33/0.05	0.37/0.05	0.41/0.04	0.43/0.04
nmsc	0.19	0.22/0.03	0.25/0.04	0.28/0.04	0.32/0.05	0.36/0.04	0.40/0.04	0.41/0.04
quadrc	0.21	0.32/0.04	0.38/0.04	0.41/0.04	0.44/0.05	0.47/0.05	0.49/0.05	0.49/0.05
qdc	0.17	0.29/0.04	0.35/0.04	0.39/0.05	0.42/0.05	0.44/0.05	0.46/0.05	0.46/0.05
udc	0.25	0.27/0.03	0.30/0.04	0.33/0.04	0.36/0.04	0.40/0.04	0.44/0.04	0.45/0.04
klldc	0.15	0.22/0.04	0.27/0.05	0.32/0.05	0.35/0.05	0.38/0.05	0.41/0.05	0.41/0.04
pcldc	0.15	0.22/0.04	0.27/0.05	0.32/0.05	0.35/0.05	0.38/0.05	0.41/0.05	0.41/0.04
knnc	**0.14**	0.18/0.03	0.22/0.04	0.26/0.05	0.31/0.05	0.36/0.05	0.40/0.04	0.41/0.04
parzenc	0.15	0.18/0.03	0.22/0.04	0.26/0.04	0.30/0.04	0.35/0.04	0.40/0.04	0.41/0.04
treec	0.22	0.27/0.03	0.31/0.04	0.35/0.04	0.39/0.04	0.43/0.04	0.46/0.03	0.46/0.03
naivebc	0.16	0.19/0.03	0.22/0.04	0.26/0.04	0.30/0.05	0.35/0.04	0.39/0.04	0.41/0.04
perlc	0.19	0.23/0.05	0.27/0.06	0.31/0.06	0.34/0.06	0.39/0.07	0.43/0.07	0.44/0.07
rbnc	0.33	0.37/0.03	0.40/0.03	0.42/0.03	0.43/0.03	0.45/0.03	0.46/0.03	0.46/0.03
svc	0.17	0.21/0.03	0.25/0.04	0.29/0.05	0.33/0.05	0.38/0.05	0.42/0.05	0.43/0.05
median	0.17	0.22/0.03	0.27/0.04	0.32/0.05	0.35/0.05	0.38/0.05	0.41/0.05	0.43/0.04

data scenario, EFC performance is close to average, which means that low probability of error for higher deficiency levels is due only to the missing data handling procedure proposed.

The results for incomplete training data have been given in Table 3.3, where '–' denotes that the classifier has not converged, and Figure 3.11. Although in this scenario the performance of the best EFC method (*efc–mimp*) is always better than the average, other classifiers can perform even better, with the difference ranging from 0.01 in most cases up to 0.03 at 90% deficiency. The performance of the remaining two EFC classifiers is always below average. Note however that Charge Redistribution outperforms casewise deletion for all deficiency levels, thus it is always beneficial to somehow take advantage of the deficient training instances rather than to discard them. Nevertheless, it appears, that for many datasets the class–conditional mean imputation performs surprisingly well. To better understand this phenomenon, the EFC decision boundaries obtained from a 2–dimensional PCA projection of the Iris dataset have been shown in Figure 3.9 alongside with a diagram generated from the same projection with maximum deficiency level and imputed mean. As it can be seen, the shape of the original decision boundaries is preserved well, and in fact even better than in the case of Charge Redistribution (see Figure 3.8(e)).

The performance estimates of all evaluated methods for the scenario with missing both test and training data have been given in Table 3.4 and Figure 3.12. The *efc–ldr/mimp* classifier steadily outperforms all other methods by a margin growing with the deficiency level.

In the semi–supervised learning scenario presented in Table 3.5 and Figure 3.13, EFC with Charge Redistribution once again performs better than average for all deficiency levels, being one of the top 3 models for the maximal level of missingness, when only two labelled instances per each class in the training data are given. Note, that EFC can work even with a single labelled instance per class, but some of the other classifiers used cannot.

The most interesting results involving semi–supervised learning from incomplete data and

Table 3.3: Performance for incomplete training data and various deficiency levels

classifier	deficiency level							
	0%	**15%**	**30%**	**45%**	**60%**	**75%**	**90%**	**max**
efc–cr	0.16	0.22/0.05	0.27/0.07	0.29/0.07	0.30/0.08	0.32/0.08	0.33/0.09	0.35/0.10
efc–mimp	0.16	0.16/0.03	0.17/0.03	0.18/0.04	0.19/0.04	0.21/0.04	0.23/0.04	0.25/0.05
efc–cwd	0.16	0.22/0.05	0.30/0.08	0.33/0.08	0.34/0.09	0.36/0.10	0.38/0.11	0.41/0.13
fisherc	0.18	0.19/0.02	0.20/0.02	0.20/0.03	0.21/0.03	0.22/0.04	0.24/0.04	0.27/0.05
ldc	0.15	**0.15/0.02**	**0.16/0.02**	**0.17/0.03**	**0.18/0.03**	**0.19/0.03**	**0.20/0.04**	**0.24/0.05**
loglc	0.15	**0.15/0.02**	0.16/0.03	**0.17/0.03**	0.19/0.04	0.21/0.04	0.24/0.05	0.27/0.06
nmc	0.23	0.23/0.02	0.23/0.02	0.23/0.03	0.24/0.03	0.24/0.03	0.25/0.04	0.26/0.04
nmsc	0.19	0.19/0.01	0.20/0.02	0.20/0.02	0.21/0.03	0.21/0.03	0.22/0.04	**0.24/0.05**
quadrc	0.21	0.23/0.02	0.23/0.02	0.24/0.03	0.26/0.03	0.28/0.04	0.30/0.05	0.32/0.08
qdc	0.17	0.19/0.02	0.19/0.02	0.20/0.03	0.21/0.04	0.22/0.05	0.26/0.06	0.34/0.08
udc	0.25	0.24/0.01	0.23/0.02	0.23/0.03	0.23/0.03	0.25/0.04	0.28/0.06	0.34/0.08
klldc	0.15	**0.15/0.02**	**0.16/0.02**	**0.17/0.03**	**0.18/0.03**	**0.19/0.03**	**0.20/0.04**	**0.24/0.05**
pcldc	0.15	**0.15/0.02**	**0.16/0.02**	**0.17/0.03**	**0.18/0.03**	**0.19/0.03**	**0.20/0.04**	**0.24/0.05**
knnc	**0.14**	0.15/0.03	0.16/0.03	0.17/0.04	0.19/0.04	0.21/0.04	0.23/0.05	0.26/0.05
parzenc	0.15	0.16/0.02	0.16/0.03	**0.17/0.03**	0.19/0.03	0.20/0.04	0.23/0.04	0.26/0.05
treec	0.22	0.23/0.04	0.26/0.05	0.27/0.06	0.29/0.06	–	–	–
naivebc	0.16	0.17/0.02	0.18/0.03	0.19/0.03	0.21/0.04	0.24/0.05	0.27/0.05	0.30/0.06
perlc	0.19	0.20/0.04	0.20/0.05	0.21/0.05	0.22/0.05	0.24/0.05	0.26/0.06	0.29/0.06
rbnc	0.33	0.35/0.03	0.37/0.03	0.38/0.03	0.40/0.03	0.42/0.03	0.44/0.03	0.47/0.03
svc	0.17	0.17/0.02	0.18/0.03	0.18/0.03	0.19/0.03	0.21/0.04	0.23/0.04	0.25/0.05
median	0.17	0.19/0.02	0.19/0.03	0.20/0.03	0.21/0.03	0.22/0.04	0.24/0.04	0.27/0.05

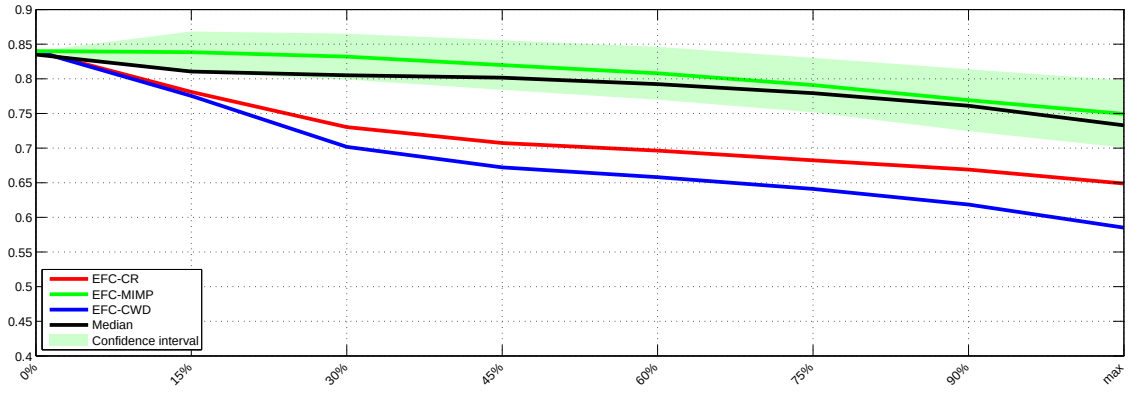


Figure 3.11: Performance for incomplete training data and various deficiency levels (recognition rates)

classification of incomplete test data have been given in Table 3.6 and Figure 3.14. As shown, the electrostatic field approach has consistently outperformed all other methods for all tested deficiency levels, by the margin reaching up to 0.06.

3.7 Concluding remarks

The Electrostatic Field Classification Framework derived in this chapter allows to address the missing data problem in an efficient and elegant manner. Apart from handling incomplete test and training data, the Framework also supports semi–supervised learning from a mixture of

Table 3.4: Performance for incomplete both test and training data and various deficiency levels

classifier	deficiency level							
	0%	**15%**	**30%**	**45%**	**60%**	**75%**	**90%**	**max**
efc–ldr/cr	0.16	0.23/0.05	0.30/0.07	0.33/0.08	0.36/0.08	0.40/0.08	0.44/0.09	0.45/0.09
efc–ldr/mimp	0.16	**0.17/0.04**	**0.20/0.04**	**0.23/0.05**	**0.27/0.06**	**0.32/0.06**	**0.36/0.07**	**0.39/0.07**
efc–ldr/cwd	0.16	0.19/0.04	0.24/0.05	0.28/0.05	0.33/0.05	0.38/0.05	0.43/0.05	0.44/0.05
fisherc	0.18	0.23/0.04	0.26/0.05	0.30/0.05	0.33/0.05	0.38/0.06	0.42/0.06	0.43/0.05
ldc	0.15	0.20/0.04	0.24/0.05	0.28/0.05	0.32/0.05	0.37/0.05	0.41/0.05	0.43/0.05
loglc	0.15	0.20/0.04	0.25/0.05	0.29/0.06	0.33/0.06	0.38/0.06	0.42/0.07	0.43/0.06
nmc	0.23	0.25/0.03	0.27/0.05	0.30/0.05	0.34/0.05	0.38/0.05	0.43/0.05	0.44/0.05
nmsc	0.19	0.22/0.03	0.25/0.04	0.28/0.05	0.32/0.05	0.37/0.05	0.41/0.05	0.43/0.05
quadrc	0.21	0.28/0.04	0.33/0.05	0.37/0.06	0.43/0.07	0.48/0.08	0.52/0.08	0.51/0.10
qdc	0.17	0.24/0.04	0.29/0.05	0.33/0.06	0.38/0.07	0.44/0.08	0.50/0.08	0.53/0.09
udc	0.25	0.27/0.03	0.29/0.05	0.32/0.05	0.37/0.06	0.42/0.07	0.49/0.08	0.53/0.09
klldc	0.15	0.20/0.04	0.24/0.05	0.28/0.05	0.32/0.05	0.37/0.05	0.41/0.05	0.43/0.05
pcldc	0.15	0.20/0.04	0.24/0.05	0.28/0.05	0.32/0.05	0.37/0.05	0.41/0.05	0.43/0.05
knnc	**0.14**	0.18/0.04	0.22/0.05	0.27/0.05	0.32/0.05	0.38/0.05	0.43/0.05	0.45/0.04
parzenc	0.15	0.18/0.04	0.22/0.04	0.27/0.05	0.32/0.05	0.38/0.05	0.43/0.05	0.44/0.05
treec	0.22	–	–	–	–	–	–	–
naivebc	0.16	0.19/0.04	0.24/0.05	0.29/0.05	0.35/0.06	0.41/0.06	0.45/0.06	0.48/0.07
perlc	0.19	0.23/0.05	0.27/0.06	0.30/0.06	0.33/0.06	0.37/0.07	0.41/0.07	0.43/0.07
rbnc	0.33	0.38/0.03	0.41/0.04	0.44/0.04	0.46/0.03	0.49/0.03	0.51/0.03	0.52/0.04
svc	0.17	0.21/0.04	0.24/0.05	0.28/0.05	0.32/0.05	0.37/0.06	0.41/0.05	0.42/0.05
median	0.17	0.21/0.04	0.25/0.05	0.29/0.05	0.33/0.05	0.38/0.06	0.43/0.05	0.44/0.05

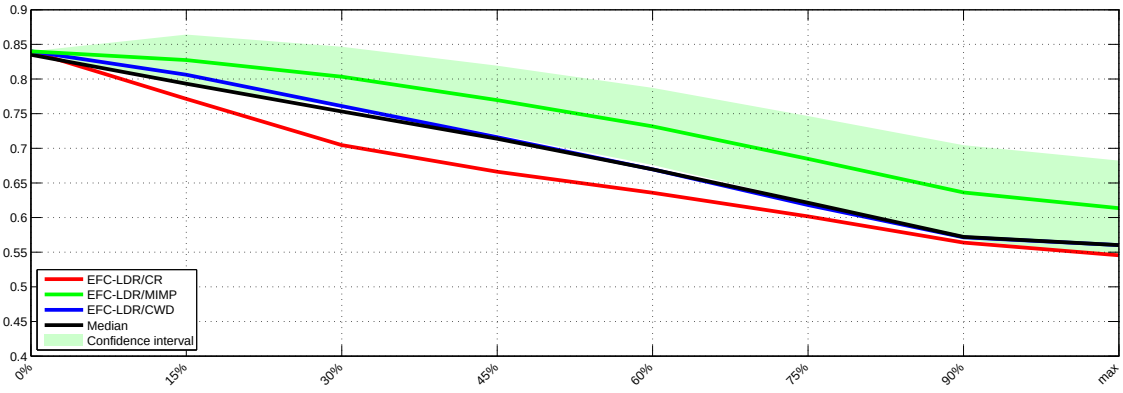


Figure 3.12: Performance for incomplete both test and training data and various deficiency levels (recognition rates)

labelled and unlabelled instances, effectively dealing with all forms of data incompleteness.

The modular, extendible architecture of the framework has allowed to develop a set of plug–in methods for dealing with different types of data deficiency as well as to integrate many of the traditional missing data handling procedures. The architecture has also allowed to easily apply the resolutions of some issues of the Electrostatic Field Classifier, which have not been addressed in previous work, including the distance concentration phenomenon in high–dimensional spaces and excess of repelling force in the generated field.

The results of the experiments performed using a number of publicly available benchmark datasets and a set of standard classification and missing data handling techniques, has lead to

Table 3.5: Performance for missing labels and various deficiency levels

classifier	deficiency level							
	0%	**15%**	**30%**	**45%**	**60%**	**75%**	**90%**	**max**
efc–gfc	0.16	0.18/0.03	0.20/0.04	0.22/0.05	0.25/0.05	0.28/0.06	0.33/0.08	0.30/0.08
efc–cr	0.16	0.16/0.02	0.17/0.03	0.17/0.04	0.19/0.04	0.21/0.05	0.24/0.06	0.30/0.08
efc–pc	0.16	0.17/0.02	0.17/0.02	0.18/0.03	0.20/0.04	0.22/0.05	0.26/0.06	0.32/0.09
fisherc	0.18	0.19/0.02	0.20/0.02	0.21/0.03	0.22/0.03	0.24/0.04	0.28/0.07	0.38/0.11
ldc	0.15	**0.15/0.01**	**0.16/0.02**	**0.16/0.02**	**0.17/0.03**	**0.18/0.03**	**0.22/0.06**	0.33/0.10
loglc	0.15	0.15/0.02	0.16/0.03	0.17/0.03	0.18/0.04	0.21/0.05	0.30/0.08	0.39/0.11
nmc	0.23	0.24/0.02	0.25/0.02	0.26/0.03	0.26/0.03	0.27/0.03	0.28/0.04	0.31/0.07
nmsc	0.19	0.20/0.02	0.21/0.02	0.21/0.02	0.22/0.03	0.23/0.03	0.24/0.04	0.29/0.08
quadrc	0.21	0.23/0.02	0.24/0.02	0.25/0.03	0.27/0.04	0.27/0.08	0.33/0.11	0.40/0.13
qdc	0.17	0.16/0.02	0.18/0.03	0.22/0.05	0.27/0.05	0.32/0.05	0.47/0.10	0.60/0.12
udc	0.25	0.24/0.02	0.24/0.02	0.25/0.03	0.25/0.03	0.26/0.03	0.28/0.06	0.40/0.13
klldc	0.15	**0.15/0.01**	**0.16/0.02**	**0.16/0.02**	**0.17/0.03**	**0.18/0.03**	**0.22/0.06**	0.33/0.10
pcldc	0.15	**0.15/0.01**	**0.16/0.02**	**0.16/0.02**	**0.17/0.03**	**0.18/0.03**	**0.22/0.06**	0.33/0.10
knnc	**0.14**	0.15/0.02	0.16/0.03	0.17/0.03	0.18/0.04	0.20/0.04	0.25/0.06	0.33/0.10
parzenc	0.15	0.16/0.02	**0.16/0.02**	0.17/0.03	0.18/0.03	0.20/0.04	0.24/0.06	0.31/0.08
treec	0.22	0.23/0.04	0.24/0.04	0.26/0.05	0.28/0.06	0.31/0.07	0.39/0.09	0.62/0.02
naivebc	0.16	0.17/0.02	0.18/0.02	0.19/0.03	0.20/0.03	0.21/0.04	0.24/0.05	**0.29/0.07**
perlc	0.19	0.19/0.04	0.20/0.05	0.20/0.05	0.21/0.05	0.22/0.06	0.26/0.07	0.36/0.11
rbnc	0.33	0.36/0.02	0.37/0.03	0.38/0.03	0.40/0.03	0.42/0.04	0.47/0.05	0.59/0.09
svc	0.17	0.17/0.02	0.18/0.02	0.18/0.03	0.19/0.03	0.20/0.04	0.24/0.06	0.32/0.09
median	0.17	0.17/0.02	0.18/0.02	0.19/0.03	0.20/0.03	0.22/0.04	0.26/0.06	0.33/0.10

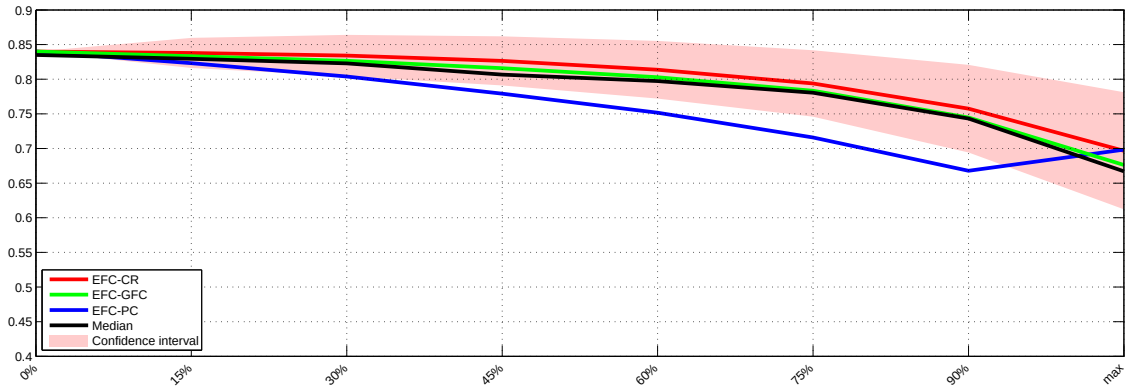


Figure 3.13: Performance for missing labels and various deficiency levels (recognition rates)

a number of interesting conclusions:

1. Even a simple direct analogy to a physical phenomenon allows to build a well–performing model able to compete with other classifiers. This confirms the potential lying in application of physical models to machine learning.

2. Inherent support for incomplete data in most cases allows to achieve better performance than imputation methods, which first try to recover the missing values and than feed the imputed dataset to a standard classification algorithm. This has an important implication, as not all standard classification algorithms can be easily extended to handle missing data. Thus in some cases imputation is the only option, leading to suboptimal

Table 3.6: Performance for missing labels, incomplete test and training data and various deficiency levels

classifier	deficiency level							
	0%	**15%**	**30%**	**45%**	**60%**	**75%**	**90%**	**max**
efc–ldr/cr/cr	0.16	0.24/0.06	0.31/0.07	0.34/0.08	0.37/0.08	0.41/0.09	0.45/0.09	**0.46/0.09**
efc–ldr/mimp/cr	0.16	**0.18/0.04**	**0.21/0.05**	**0.25/0.05**	**0.29/0.06**	**0.35/0.07**	**0.41/0.08**	**0.46/0.09**
fisherc	0.18	0.24/0.04	0.28/0.05	0.32/0.05	0.37/0.06	0.42/0.06	0.47/0.06	0.50/0.07
ldc	0.15	0.20/0.04	0.25/0.05	0.29/0.05	0.35/0.06	0.40/0.06	0.46/0.06	0.49/0.08
loglc	0.15	0.21/0.04	0.26/0.05	0.31/0.06	0.36/0.07	0.42/0.07	0.47/0.08	0.48/0.08
nmc	0.23	0.26/0.04	0.29/0.05	0.32/0.05	0.36/0.06	0.42/0.06	0.46/0.06	0.48/0.06
nmsc	0.19	0.23/0.03	0.26/0.05	0.31/0.05	0.35/0.05	0.40/0.06	0.46/0.06	0.50/0.08
quadrc	0.21	0.30/0.04	0.36/0.06	0.42/0.06	0.47/0.08	0.51/0.09	0.53/0.09	0.51/0.07
qdc	0.17	0.25/0.05	0.32/0.06	0.39/0.07	0.46/0.08	0.51/0.07	0.60/0.09	0.63/0.10
udc	0.25	0.27/0.04	0.30/0.05	0.35/0.06	0.41/0.07	0.48/0.08	0.58/0.10	0.63/0.10
klldc	0.15	0.20/0.04	0.25/0.05	0.29/0.05	0.35/0.06	0.40/0.06	0.46/0.06	0.49/0.08
pcldc	0.15	0.20/0.04	0.25/0.05	0.29/0.05	0.35/0.06	0.40/0.06	0.46/0.06	0.49/0.08
knnc	**0.14**	0.19/0.04	0.24/0.05	0.30/0.05	0.36/0.06	0.43/0.05	0.47/0.05	0.49/0.06
parzenc	0.15	0.19/0.04	0.24/0.05	0.30/0.05	0.36/0.06	0.42/0.05	0.46/0.05	0.49/0.06
treec	0.22	0.28/0.05	–	–	–	–	–	0.58/0.06
naivebc	0.16	0.20/0.04	0.25/0.05	0.32/0.06	0.38/0.06	0.44/0.07	0.49/0.08	0.53/0.09
perlc	0.19	0.24/0.05	0.27/0.06	0.31/0.06	0.35/0.07	0.40/0.07	0.48/0.09	0.54/0.09
rbnc	0.33	0.40/0.03	0.44/0.03	0.47/0.03	0.49/0.03	0.51/0.03	0.55/0.06	0.63/0.10
svc	0.17	0.21/0.04	0.25/0.05	0.30/0.05	0.35/0.06	0.40/0.06	0.45/0.06	0.47/0.07
median	0.17	0.23/0.04	0.26/0.05	0.31/0.05	0.36/0.06	0.42/0.06	0.47/0.06	0.49/0.08

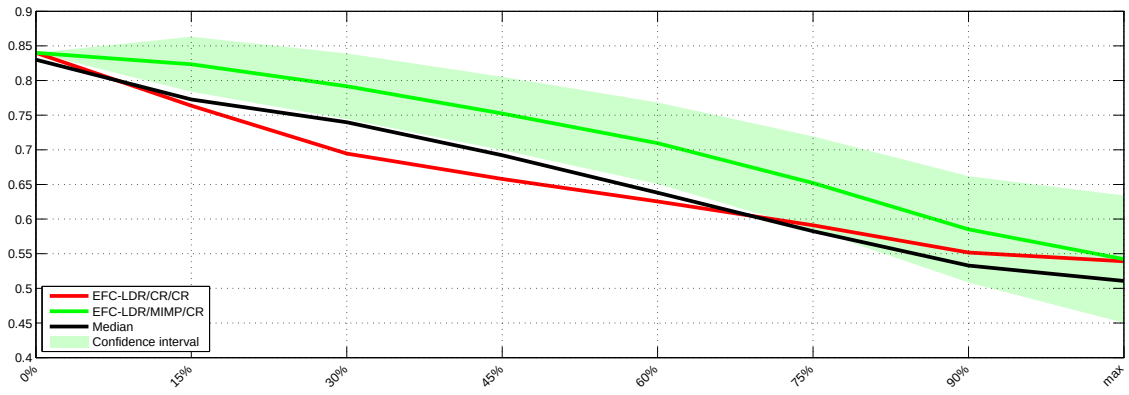


Figure 3.14: Performance for missing labels, incomplete both test and training data and various deficiency levels (recognition rates)

classification performance. More specifically, the experiments have revealed that:

- The EFC–specific Local Dimensionality Reduction technique for processing incomplete test data is superior to all other tested classifiers combined with the class–conditional mean imputation technique.
- The EFC–specific Charge Redistribution method has allowed EFC to become one of the top performing solutions in the semi–supervised learning scenario.
- The Charge Redistribution method however did not perform so well in the case of missing training data, yet the Electrostatic Field Classifier used together with the class–conditional mean imputation is still performing better than average.

- In the semi–supervised learning scenario with deficient both test and training data, EFC has once again outperformed all other classifiers.

From the results presented, there also emerges a clear pattern of which EFC approach to missing data is most likely to produce the best performance within a particular missing data scenario. The following practical guideline to using EFC in various incomplete data situations has been closely followed in the practical part of this thesis described in Chapter 6.

- Incomplete test data – Local Dimensionality Reduction,
- Incomplete training data (missing features) – Class–conditional Mean Imputation,
- Incomplete label data – Charge Redistribution.

As shown in the experiments, the combination of the above methods is able to provide good recognition rates for the most difficult scenarios with missing features and labels, even for high deficiency levels, in which case the proposed method not only performs better than average, but in fact often is one of the top performing models.

Chapter 4

Density Preserving Sampling for error estimation and model selection

4.1 Introduction

Estimation of the generalisation ability of a classification or regression model is an important issue in the machine learning field, especially that it is independent of the actual model used. Generalisation accuracy estimates are not only used as indicators of the expected performance of the developed classifier or regressor on previously unseen data, but are also commonly used for model ranking and selection [40].

In contrast to the large number of various regression and classification methods currently in use, there is only a handful of model independent generalisation error estimation techniques. The most popular of them are cross–validation [29] dating back to 1968, and bootstrap [44] developed in the 1979. These techniques, and especially cross–validation are being used even more willingly and blindly after the publication of a seminal paper by Kohavi in 1995, presenting a comparative study of bootstrap and cross–validation [92], and currently estimated to have more than 1700 direct citations according to Harzing's Publish or Perish citation retrieval system[1].

The basic idea, shared by all generalisation error estimation methods, is to reserve a subset of available data to test the model after it has been trained using the remainder of the dataset. The main difference between various techniques is the way the generalisation error is calculated, the size of the subset reserved for testing or whether the procedure is repeated multiple times or not. They have however something in common, and that is the way in which the testing subset is generated – random sampling. Although the stochastic nature of bootstrap and cross–validation ensures that in the limit they would both converge to a true value, this may also lead to large variations in the estimate between consecutive runs, making the results unreliable. This effect can be alleviated to a large extent by repeating both procedures multiple times, which however considerably increases the computational demands.

A good test set should be independent of the training data and representative of the population from which it has been drawn. While random sampling meets the first requirement, it does not guarantee the representativeness of the test set. In order to address this issue, stratified sampling approaches have been developed [92], which try to increase the representativeness at the expense of independence, and are able to achieve better results than their non–stratified counterparts.

Inspired by the success of stratified sampling approaches, a Density Preserving Sampling

[1] http://www.harzing.com/pop.htm

procedure is proposed here, which further sacrifices the independence of the test set to enforce its representativeness. The method achieves this goal by optimising the correntropy, a recently developed, non–parametric similarity measure of the probability density functions [102], and as shown in the experimental section produces accurate generalisation estimates requiring a fraction of computations when compared to cross–validation.

4.2 Generalisation error estimation

Generalisation error is the error a predictive model will make on novel, previously unseen data, generated from the same distribution as the data used to develop the model [40]. Low generalisation error is thus a sign of a good match between the model and the problem, and lack of overfitting [11].

It is impossible to obtain a closed form solution for calculation of the generalisation error or even for calculation of tight bounds for the error, in all but the simplest cases [5]. The only practical solution is to estimate the generalisation error from all available i.i.d. (independently and identically distributed) data instances by splitting them into training and validation sets [81]. For the error estimate to be meaningful both these datasets should be representative of the true distribution, so the way in which the data is split plays a crucial role.

4.2.1 Hold–out and random subsampling

The simplest and the least computationally expensive way to estimate the generalisation error is the hold–out method [181], in which the data is split randomly into two parts: the training set and the hold–out set, in a priori chosen proportions. This has been depicted in Figure 4.1. It is a common practice for most sampling techniques to keep the training set bigger than the test set. The rationale behind this approach is that the test set is only used to determine a single property (e.g. when to stop training or which model to choose), while the training set is used to adjust possibly hundreds or more parameters of the model [40].

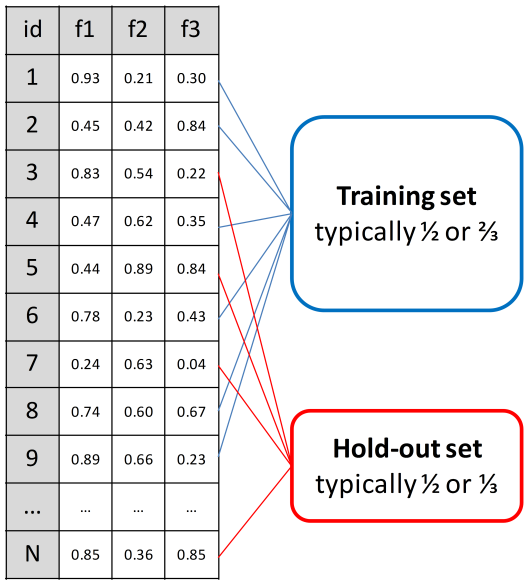


Figure 4.1: Hold–out method; ‘f1’,‘f2’,‘f3’ denote features, ‘id’ denotes instance indexes

The model is then trained using the training dataset and its error on the hold–out data becomes an estimate of the generalisation error. The obvious drawback of the hold–out method is that unless both datasets are large enough (which is a vague term by itself), different estimates will be obtained from one run to another. A workaround known as random subsampling [81], repeats the hold–out procedure multiple times and averages the results. This procedure however still has some disadvantages as it does not guarantee that all instances will at some point be used for training nor that none of the classes will be over/under–represented in the hold–out set [92]. In order to circumvent these issues, more advanced resampling techniques have been developed. Yet, the hold–out method is still being used when dealing with large datasets, as in this case other techniques quickly become untractable. It is also sometimes assumed that more advanced resampling techniques are simply not needed for large amounts of data.

4.2.2 Cross–validation

Cross–validation is a widely used standard statistical technique for estimation of model generalisation ability, applied with a great success to both classification and regression problems [11, 40]. In k–fold cross–validation the whole available dataset is first randomly divided into k approximately equal subsets. This has been depicted in Figure 4.2. Each of these subsets or folds is then in turn put aside as validation data, a model is built using the remaining $k-1$ subsets and tested using the validation subset. The estimate of the generalisation error is then calculated as a mean value of all validation errors, while the standard deviation of the validation error can be used to approximate the confidence intervals of obtained error estimate. The whole procedure thus requires development of exactly k models. Since the results obtained in the setting described above are also likely to vary from one run to another, the procedure is usually repeated multiple times for various random splits, and the results are averaged.

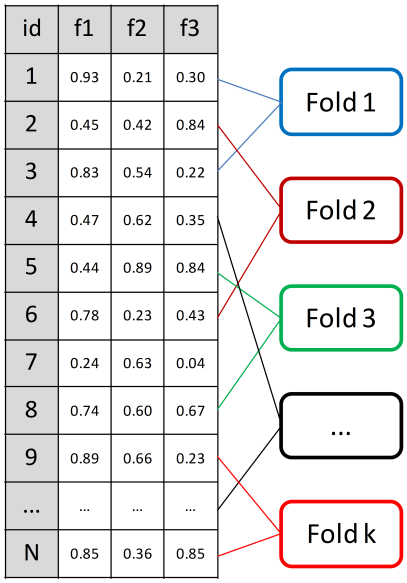


Figure 4.2: Cross–validation; ‘f1’,‘f2’,‘f3’ denote features, ‘id’ denotes instance indexes

The most often used variants of cross–validation are:

- Leave–one–out (LOO) cross–validation in which a single instance is used as a validation

set each time. This produces unbiased error estimates but with high variance and can be computationally prohibitive for large datasets.

- Repeated 10–fold cross–validation, which often is a good compromise between speed and accuracy.
- Repeated 2–fold cross–validation, which is an approximation of the bootstrap method [181].

In order to improve the accuracy of the estimates obtained, a stratified cross–validation approach is used in practice, which samples the data in a way that approximates the percentage of each class in every fold [92]. For regression problems, stratified cross–validation produces folds with equal mean values of the target variable [81].

4.2.3 Bootstrap

Bootstrap is a second commonly used generalisation error estimation procedure [11, 40], especially useful when dealing with small datasets [181]. Given an input dataset of size N, the method performs uniform sampling with replacement to produce a training set of the same size. The instances not picked during the sampling procedure form the test set. This has been depicted in Figure 4.3. The probability of each instance ending up in the test set is thus given by:

$$\left(1 - \frac{1}{N}\right)^N \approx e^{-1} \approx 0.368 \tag{4.1}$$

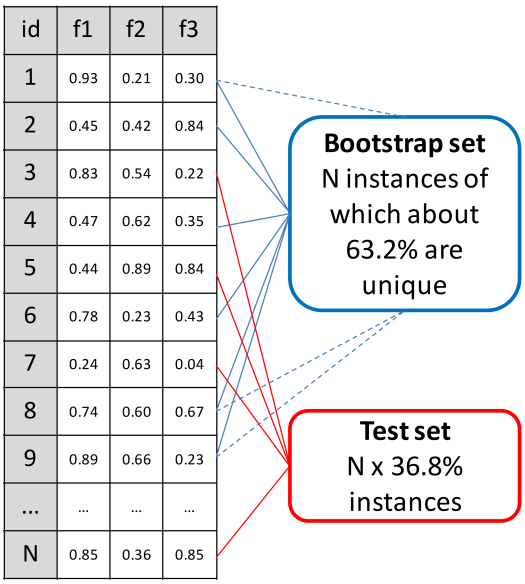


Figure 4.3: Bootstrap method; ‘f1’,‘f2’,‘f3’ denote features, ‘id’ denotes instance indexes

Due to the fact that the probability of each instance being picked for training is $1 - 0.368 = 0.632$, the method is also often called the ‘0.632 bootstrap’ [181]. Since the error estimate obtained using test data only would be overly pessimistic (only about 63.2% of instances are used for training every time), the generalisation error estimate is calculated using weighted test

and training errors:

$$error = \frac{1}{b}\sum_{i=1}^{b}(0.632 \times e_{test} + 0.368 \times e_{all}) \tag{4.2}$$

where b is the number of bootstrap samples and e_{all} is the error on all instances from the original dataset. In general, the more times the whole process is repeated, the more accurate the estimate. A comprehensive comparative study of cross–validation and bootstrap has been described in [92].

4.2.4 Bias and variance of error estimation methods

The bias of an error estimation method is the difference between the expected value of the error and the estimated value [92]. For an unbiased estimator, this difference is equal to zero. Bias can also be either positive or negative. In the former case, the estimate is said to be overly optimistic, as the estimated error is lower than the expected error. Negative bias on the other hand leads to overly pessimistic error estimates.

Low bias on its own does not guarantee good performance of the model. There is another important parameter – the variance, which measures the variability of the error estimate from one run to another. In the case of subsampling methods discussed in this chapter, the variability is usually approximated by the expected standard deviation of a single accuracy estimation run [92]. A good generalisation error estimator should thus have low bias and low variance. Unfortunately in practice it is usually difficult to achieve both at the same time, hence this situation can be perceived as another facet of the bias–variance dilemma discussed in Section 2.2.2.

4.3 Information Theoretic Learning for entropy manipulation

As discussed in Section 2.3.2, Information Theoretic Learning is a procedure of adapting the parameters of a learning machine using information theoretic criterion [127]. However due to the omnipresent 'learning from exemplars' paradigm, application of the information theory to learning problems is not straightforward, mainly because the required parametric forms of the PDFs rarely exist. The ITL framework developed in [127] effectively addresses this issue by using alternative definitions of the information theoretic measures, which are easier to estimate. For more details on the ITL framework please refer to Appendix C.

4.3.1 Renyi's quadratic entropy

Entropy is a measure of the uncertainty associated with a random variable. It quantifies the average information content that is missing due to the unknown value of the variable and is the main criterion used in ITL approaches. The higher the entropy, the more information can be gained by discovering the value of the variable.

Calculation of Shannon's entropy requires the density function to be given in an analytic form. There are however other definitions of entropy that can be used in the ITL framework and Renyi's entropy is one of them. The definition of Renyi's entropy of order α for a continuous random variable is given by:

$$H_{R_\alpha}(Y) = \frac{1}{1-\alpha}\log\int p(y)^\alpha \, dy \tag{4.3}$$

Renyi's entropy involves calculation of the integral of the power of PDF rather than integral of the logarithm as in the case of Shannon's counterpart, which is much easier to estimate [128].

Moreover, Shannon's entropy is the limiting case of Renyi's entropy when $\alpha \rightarrow 1$. For practical applications the choice of $\alpha = 2$ is a good compromise between robustness and computational complexity ($O(n^2)$) [128], which leads to the definition of Renyi's quadratic entropy:

$$H_{R_2}(Y) = -\log \int p(y)^2 \, dy \tag{4.4}$$

The most important property of Renyi's entropy from the point of view of ITL is that the extrema of H_S and H_R overlap [127], so both definitions are equivalent for the purpose of entropy optimisation.

The above criterion is however still useless without a good estimate of the probability density function, which fortunately can be obtained and efficiently integrated into Eq. 4.4 by using the Parzen window density estimator [40]. Denoting by $G(\boldsymbol{y},\ \sigma^2\boldsymbol{I})$ a spherical Gaussian kernel centered at $\boldsymbol{y}$ with a diagonal covariance matrix $\sigma^2\boldsymbol{I}$, the PDF can be estimated as follows:

$$\hat{p}(\boldsymbol{y}) = \frac{1}{N}\sum_{i=1}^{N} G(\boldsymbol{y} - \boldsymbol{y}_i,\ \sigma^2\boldsymbol{I}) \tag{4.5}$$

Substituting Eq. 4.5 into Eq. 4.4 and using the convolution property of the Gaussian kernel, i.e. $\int G(\boldsymbol{z} - \boldsymbol{y}_i, \sigma_1^2\boldsymbol{I})G(\boldsymbol{z} - \boldsymbol{y}_j, \sigma_2^2\boldsymbol{I})\, d\boldsymbol{z} = G(\boldsymbol{y}_i - \boldsymbol{y}_j, \sigma_1^2\boldsymbol{I} + \sigma_2^2\boldsymbol{I})$, yields:

$$H_{R_2}(Y) = -\log \int \hat{p}(\boldsymbol{y})^2 \, d\boldsymbol{y} = -\log V(\boldsymbol{y}) \tag{4.6}$$

$$\begin{aligned} V(\boldsymbol{y}) &= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} \int G(\boldsymbol{z} - \boldsymbol{y}_i,\ \sigma^2\boldsymbol{I})\ G(\boldsymbol{z} - \boldsymbol{y}_j,\ \sigma^2\boldsymbol{I})\ d\boldsymbol{z} \\ &= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} G(\boldsymbol{y}_i - \boldsymbol{y}_j,\ 2\sigma^2\boldsymbol{I}) \end{aligned} \tag{4.7}$$

Renyi's entropy of order α calculates the interactions between α–tuplets of instances, so the higher the value of α, the more information about the structure of the dataset can be extracted [128] but the computational complexity – $O(n^\alpha)$ – quickly becomes prohibitive.

If some imaginary particles were placed on top of each data instance a potential field would be created, since $G(\boldsymbol{y}_i - \boldsymbol{y}_j,\ 2\sigma^2\boldsymbol{I})$ is always positive and decays exponentially with the square of the distance between $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ [127]. The instances can thus be referred to as Information Particles while $V(\boldsymbol{y})$, which is an averaged sum of all pairs of interactions and represents the total potential energy of the dataset – Information Potential (IP). By analogy to classical physics the gradient of potential energy is a force, which would drag the particles to a state with minimum potential if they were free to move. This behaviour can be used for training of adaptive systems with forces taking place of the injected error and used for adjusting parameters of the model [128].

4.3.2 Auto and cross correntropy

A Generalised Correlation Function (GCF) for a stochastic process T is defined as [148]:

$$V_T(T_1, T_2) = E[\phi(t_1), \phi(t_2)] = E[K(t_1, t_2)] \tag{4.8}$$

where E stands for the expected value, ϕ denotes some kernel induced transformation and K is a kernel function, assumed to be Gaussian from now on. It has been proven, that the GCF estimator not only conveys information about autocorrelation but also about the structure of the dataset, as its mean value for non–zero lags converges asymptotically to the estimate of the Information Potential calculated using Renyi's quadratic entropy [148]. For this reason the function has been named auto–correntropy and is a preferred choice over traditional methods also due to taking advantage of all even moments of the PDF.

The idea of auto–correntropy has been further developed in [102] for a general case of two arbitrary random variables. The new measure, named cross–correntropy (or correntropy) is defined for variables X and Y as:

$$V_{XY}(X,Y) = E[\phi(x), \phi(y)] = E[K(x,y)] \tag{4.9}$$

The correntropy can be used as a measure of similarity between X and Y but only in the neighbourhood of the joint space. This results from the restriction of Gaussian kernels, which have high values only along the $x \approx y$ line with exponential fall off otherwise. The size of this neighbourhood is therefore controlled by the kernel width parameter σ. As a result, correntropy can also be defined as the integral of the joint probability density along the line $x = y$:

$$V_{XY}(X,Y) \approx \int p(x,y) \mid_{x=y=u} du \tag{4.10}$$

The joint PDF can be estimated from the data using the Parzen window method:

$$p(\boldsymbol{x}, \boldsymbol{y}) \approx \frac{1}{N} \sum_{i=1}^{N} G(\boldsymbol{x} - \boldsymbol{x}_i, \sigma^2 \boldsymbol{I}) G(\boldsymbol{y} - \boldsymbol{y}_i, \sigma^2 \boldsymbol{I}) \tag{4.11}$$

By integrating the above along the $x = y$ line and using the convolution property of Gaussian functions again, the estimate of correntropy is finally obtained as:

$$V_{XY}(X,Y) \approx \frac{1}{N} \sum_{i=1}^{N} G(\boldsymbol{x}_i - \boldsymbol{y}_i, 2\sigma^2 \boldsymbol{I}) \tag{4.12}$$

The correntropy can thus be regarded as the PDF of equality of two variables in the neighbourhood of the joint space, of the size determined by the kernel width parameter σ [102, 104]. The measure has many interesting properties and one of them is that for independent X and Y it can be approximated by the Information Potential formula similar to Eq. 4.7 and named Cross Information Potential [104]. Correntropy has been successfully used as a localized, outlier–resistant similarity measure for various supervised learning applications [86, 103, 104, 154].

4.4 Density Preserving Sampling procedure

Both cross–validation and bootstrap, described in Sections 4.2.2 and 4.2.3 are stochastic methods. The immediate consequence is that the results can vary a lot from one run to another and there is no guarantee that the datasets obtained by splitting the original data are representative, which is a necessary condition for obtaining accurate error estimates. For this reason, in order to obtain reliable results, averaging over multiple iterations is required. In general, the more

times the procedure is repeated the better, as in the limit both methods will converge to the true error values. For $k-$fold cross–validation using $N-$element dataset this could mean averaging over all $\binom{N}{N/k}$ possibilities of choosing N/k instances out of N (the so called 'complete cross–validation' [92]), which quickly becomes untractable. There is however another, often overlooked possibility – intelligent sampling aiming at producing only representative splits.

From statistics, a random sample is considered representative if its characteristics reflect those of the population from which it is drawn [37]. Since these characteristics are reflected by the probability density function, the more similar the distribution of the sample to the distribution of the population, the more representative this sample is. The correntropy described in Section 4.3.2 can be used to measure the similarity between two distributions and thus to measure the 'representativeness' of the sample. Moreover, it is also possible to use correntropy as an optimisation criterion, guiding the sampling process in order to split a given dataset into two or more maximally representative subsets.

4.4.1 Estimation of correntropy for unsorted datasets with different cardinalities

Eq. 4.12 defines correntropy between two random variables or datasets X and Y as the value of a Gaussian kernel centered at $(\boldsymbol{x}_i - \boldsymbol{y}_i)$ averaged over all N instance pairs. There are thus three requirements for calculation of correntropy to be possible: the datasets (1) must be ordered, (2) must have the same dimensionality and (3) must have the same number of objects. While the second requirement is irrelevant for sampling, as each subset of objects necessarily needs to have the same dimensionality as the set from which it has been selected, the remaining two requirements may pose a problem.

For some applications like e.g. supervised learning, all the above requirements are met automatically – if X denotes the output of a mapper and Y denotes the target value, $|X| = |Y|$ and $\boldsymbol{x}_i$ is the prediction of $\boldsymbol{y}_i$. In sampling however in general one cannot expect the instances to be ordered, which means that it is not obvious on the difference of which instances to center the Gaussians. Moreover, the datasets may have different cardinalities e.g. when one wants to calculate the correntropy between the original dataset and its subset.

To address the ordering issue the following approach is adapted. For every instance $\boldsymbol{x}_i, i \in (1..N)$, the Gaussian is centred at $(\boldsymbol{x}_i - \boldsymbol{y}_j)$, such that:

$$j = \underset{j}{\operatorname{argmin}} \parallel \boldsymbol{x}_i - \boldsymbol{y}_j \parallel, j \in (1..N) \tag{4.13}$$

where $\parallel \cdot \parallel$ denotes the Euclidean distance. In other words $\boldsymbol{y}_j$ is selected to be as close to $\boldsymbol{x}_i$ as possible. Both $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$ are then removed from their respective sets and the procedure is repeated until all instances are exhausted. The generalised, instance ordering insensitive formula for calculation of correntropy thus becomes:

$$\begin{aligned} V_{XY}(X,Y) &\approx \frac{1}{N}\sum_{i=1}^{N} G(\boldsymbol{x}_i - \boldsymbol{y}_j, 2\sigma^2\boldsymbol{I}) \\ j &= \underset{j}{\operatorname{argmin}} \parallel \boldsymbol{x}_i - \boldsymbol{y}_j \parallel, \;\; j \in J_{avail} \end{aligned} \tag{4.14}$$

where the set J_{avail} contains indices of $\boldsymbol{y}$ which haven't yet been used, to ensure that each $\boldsymbol{y}_k$ is used only once.

When the datasets have different cardinalities, that is without loss of generality if $N_X > N_Y$,

the approach outlined above will terminate after N_Y instances are processed. To avoid this, a new dataset Y_N is created by duplicating the original Y dataset $\lceil N_X/N_Y \rceil$ times. Correntropy is then calculated between X and Y_N and the calculation will terminate after exactly N_X steps.

For the correntropy values to be more comparable for different experiments, $V_{XY}(X,Y)$ has been scaled to fit into the $[0,\ +1]$ range, by dividing each $G(\boldsymbol{x}_i - \boldsymbol{y}_j, 2\sigma^2\boldsymbol{I})$ in the sum in Eq. 4.14 by $G(0, 2\sigma^2\boldsymbol{I})$. Every correntropy value given in the rest of this chapter has been scaled. Note however, that the correntropies should be compared with caution as their absolute difference can be made almost arbitrarily large by manipulating the Parzen window width parameter σ. For this reason the correntropy values given should be seen as ranks of various models/solutions on an ordinal scale.

4.4.2 Correntropy based sampling procedure

In this section a correntropy–based, hierarchical, binary density preserving splitting procedure is proposed. The correntropy given by Eq. 4.12 is a function differentiable with respect to both $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$, which unfortunately is not the case for the generalised function given by Eq. 4.14. Moreover, none of them is differentiable with respect to the indices i and j, which are the only variables that can be manipulated within the splitting process. Gradient driven optimisation procedure is thus not straightforward hence a greedy, locally optimal approach was chosen.

Since correntropy is being estimated by a scaled sum of Gaussians, it reaches a maximum when all components of the sum reach their maximal values. In case of a single Gaussian function, the maximum is reached at 0 and since the function is piecewise monotonic and symmetric, the closer $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$ are in Eq. 4.14, the higher $V_{XY}(X,Y)$ will be. This immediately suggests an iterative, binary splitting procedure of a dataset Z into datasets X and Y, which at each step selects two instances $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ so that:

$$i, j = \underset{i,j}{\operatorname{argmin}} \parallel \boldsymbol{z}_i - \boldsymbol{z}_j \parallel \tag{4.15}$$

and then adds them to the sets X and Y, so that $X = X \cup \boldsymbol{z}_i$ and $Y = Y \cup \boldsymbol{z}_j$ or the other way round, removing them from dataset Z at the same time. This has been depicted in Figure 4.4.

The above procedure aims at directly maximising $V_{XY}(X,Y)$, that is the correntropy between the two new datasets. Due to the way correntropy is calculated for sets with various cardinalities however, it also indirectly maximises $V_{XZ}(X,Z)$ and $V_{YZ}(Y,Z)$. As a result, newly obtained datasets are splits with distributions maximally similar to each other and to the distributions of the original dataset in the correntropy sense. To obtain more than 2 splits, the procedure can be repeated by splitting datasets X and Y again, which will produce 4 splits and so on. The total number of splits is thus always a power of 2.

The instances $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ can be added to the sets X and Y arbitrarily or not. In the approach presented here a procedure in which the two objects are distributed in a way that maximises the average coverage of the input space by both splits has been devised. Denoting by d_{kW} the average Euclidean distance between instance $\boldsymbol{z}_k$ and all instances in set W, the rules are:

$$d_{iX} + d_{jY} \geq d_{jX} + d_{iY} \ \Rightarrow X = X \cup \boldsymbol{z}_i,\ Y = Y \cup \boldsymbol{z}_j \tag{4.16}$$

$$d_{iX} + d_{jY} < d_{jX} + d_{iY} \ \Rightarrow X = X \cup \boldsymbol{z}_j,\ Y = Y \cup \boldsymbol{z}_i \tag{4.17}$$

For classification problems, the splitting procedure can be executed in either supervised or unsupervised mode. In the former case, the algorithm takes advantage of the class labels supplied

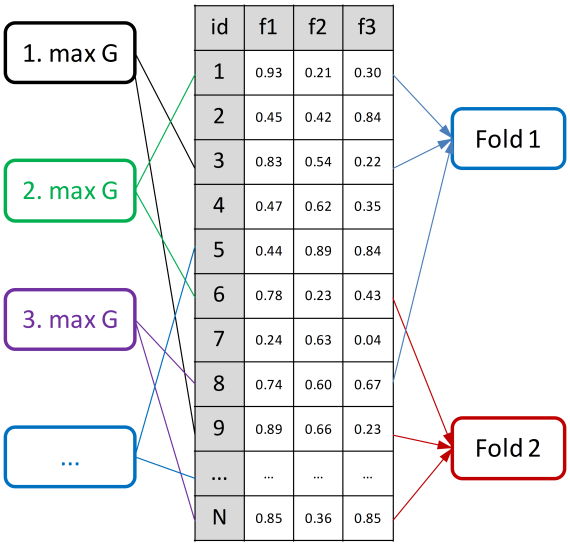


Figure 4.4: Density Preserving Sampling procedure; 'f1','f2','f3' denote features, 'id' denotes instance indexes

with the data by considering each class in turn and in separation from the rest. In other words the dataset is being split class by class. This approach is referred to as DPS–S. In the unsupervised mode, the class labels are ignored, so the procedure is purely density–driven and has been called DPS–U. Similar remark applies to estimation of correntropy, which can also be calculated in a class–wise (supervised) or class–less (unsupervised) mode.

In current implementation, if the classes are too small to be divided into a given number of subsets, DPS–S automatically falls–back to DPS–U. Since the splitting procedure is hierarchical, most datasets can be divided using the supervised approach up to some point, after which the unsupervised procedure will take over.

The computational complexity of the DPS approach is of the order $O(N^2/2)$ in the unsupervised case, as the most time consuming operation is calculation of pairwise distances between all N instances in a form of a symmetric distance matrix. For supervised DPS, the complexity is of order $O(\sum_i N_i^2/2)$, where N_i is the cardinality of the i^{th} class. This is however negligible when compared to the complexity of most training algorithms.

4.5 Experiments

The experiments have been conducted on 27 publicly available datasets using a total of 16 different classifiers. The datasets used have been described in Appendix A. For the list of classifiers used please refer to Appendix B.1.

The experiments were designed to (1) compare the error estimation accuracy of cross–validation and Density Preserving Sampling approaches, (2) test the stability of both error estimators, (3) test applicability of DPS to the classifier selection process, (4) check, if it is possible to reliably estimate the generalisation error using a single DPS fold only, thus reducing the computational requirements by another order of magnitude, and (5) examine the behaviour of DPS in the context of ensemble models, in comparison to cross–training.

An approach similar to the one outlined in [92] has been followed. For each dataset a stratified

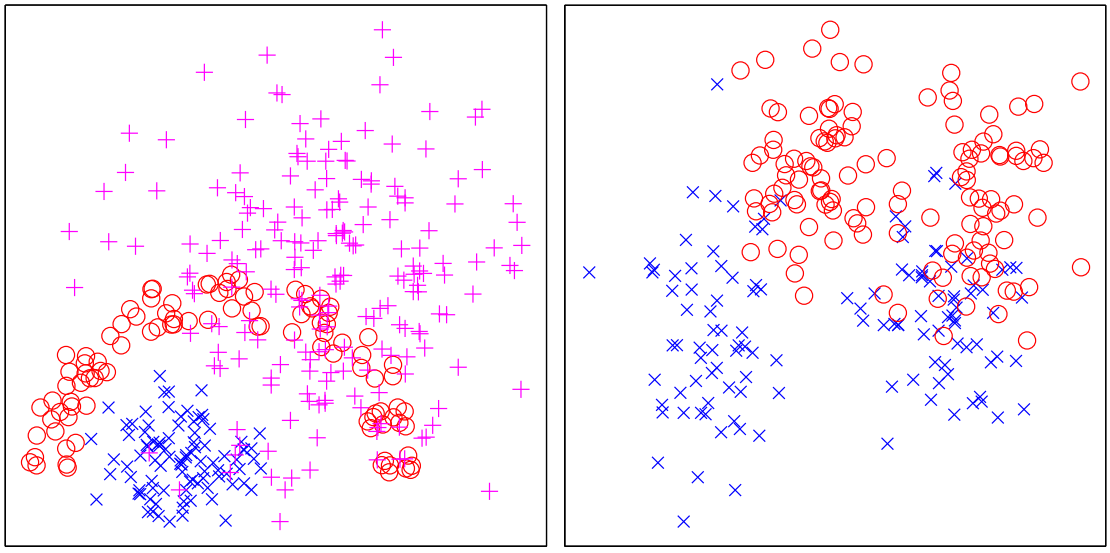
random subsampling procedure has been repeated 100 times, resulting in 100 random divisions of the dataset into a training part $(2/3)$ and independent test data $(1/3)$. The training part was then used to estimate the generalisation error using CV and DPS for each classifier, while the independent test part has been used to calculate the 'true' generalisation error, once again for each classifier in turn. The true generalisation error then served to calculate the bias of each estimate, while the generalisation error estimates of a single estimation run have been used to calculate the variance. Finally, the results have been averaged over all 100 runs of the random subsampling procedure.

The CV estimate has been calculated within a 10 times repeated 8–fold cross–validation scheme. The average results for all 10 iterations as well as the result of the best and worst single run in terms of bias/variance have been provided, in order to emphasise how wrong the things can go with CV.

Three 8–fold DPS estimates are also given – DPS–S (using class label information), DPS–U (ignoring class label information) and DPS–SU (averaged over the two).

4.5.1 Toy problems

The analysis starts with two synthetic datasets first used in [97] and [134]. The datasets have been chosen as they are both two–dimensional, which allows for visualisation of the results and have been extensively used in many previous studies due to their well known properties.



(a) Cone–torus (b) Synth–mat

Figure 4.5: Synthetic datasets

Cone–torus dataset

Cone–torus is a synthetic, 2 dimensional dataset consisting of 3 classes. A scatter plot of the dataset is given in Figure 4.5(a). Figure 4.6 depicts scatter plots of 8 DPS–S folds, while in Figure 4.7, 8 CV folds generated during a single random run are given. Note, that in case of DPS, the classes tend to preserve their shapes – the half torus for example is clearly visible in 7 out of 8 folds, while for CV only in 4 or 5. This is also well reflected by the mean value of

correntropy between all 8 folds and the original dataset, which is 0.81 for DPS and 0.71 for CV averaged over 10 runs ($\sigma = 0.12$).

The decision boundaries for the *qdc* classifier trained on each of 8 folds in turn, superimposed on the original dataset have been given in Figure 4.8. The black solid line represents the boundaries of a classifier trained using the DPS–S folds, while the blue dotted line shows the boundaries for a single CV run. Notice, that for DPS the decision boundaries generally do not change their shape from one fold to another, as opposed to CV, where the boundaries seem very unstable and can change radically.

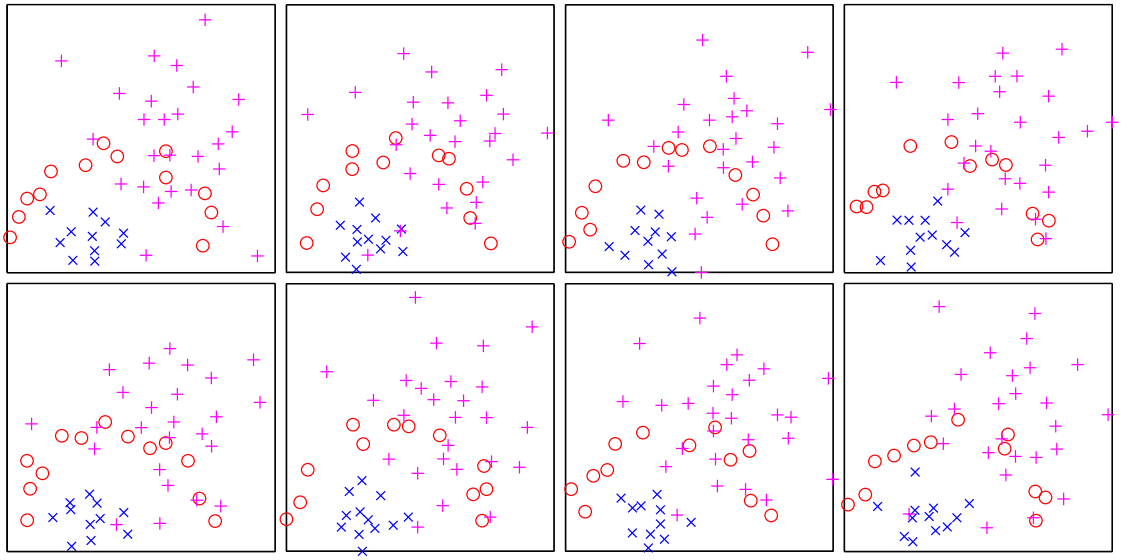


Figure 4.6: Cone–torus – scatter plots of 8 DPS–S folds

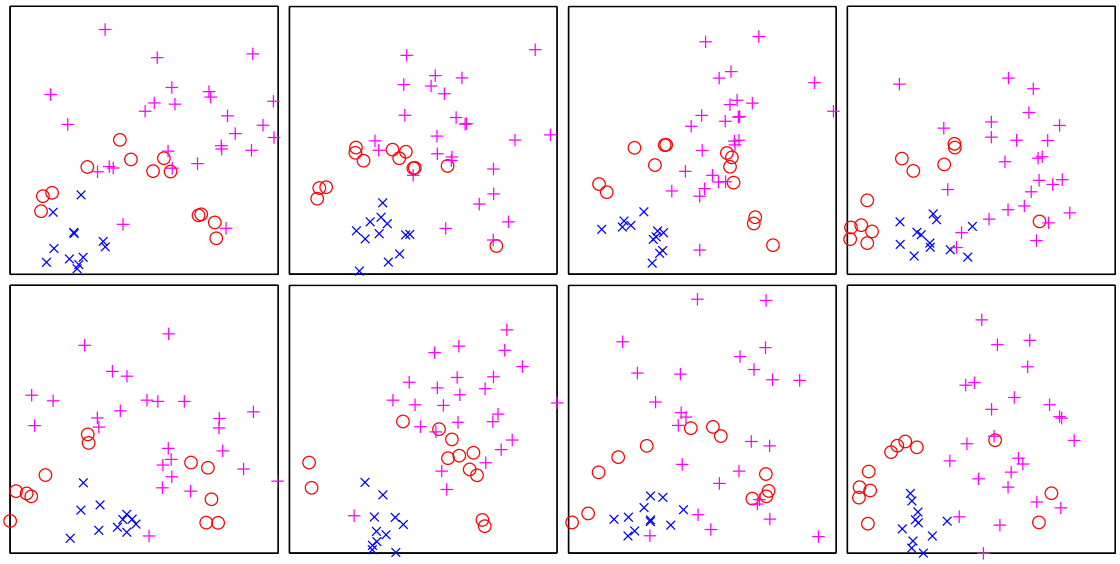


Figure 4.7: Cone–torus – scatter plots of 8 CV folds

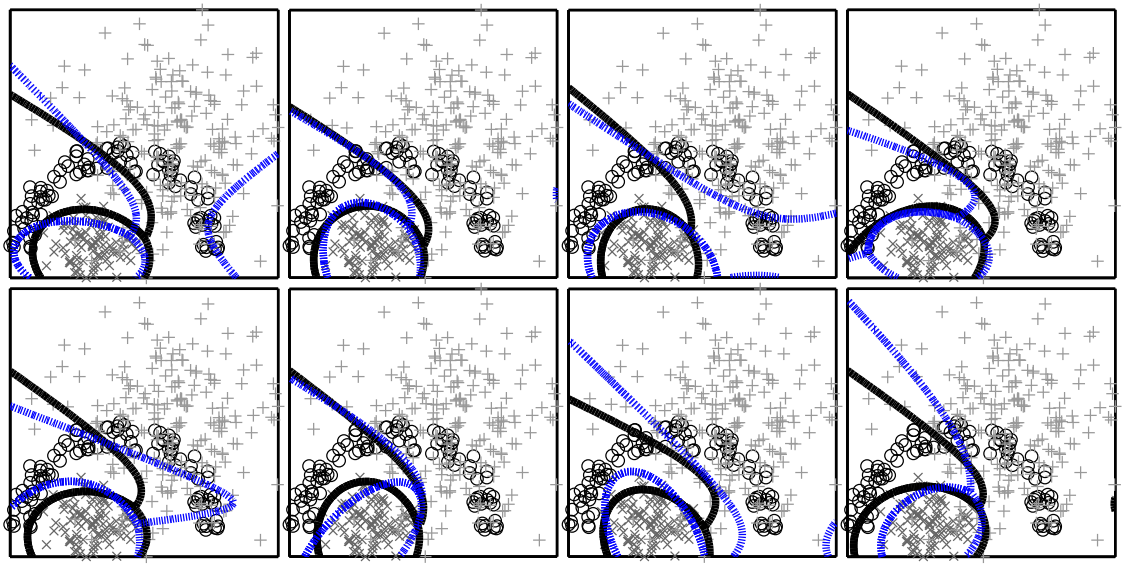


Figure 4.8: Cone–torus – decision boundaries for *qdc* trained on DPS–S (solid line) and CV (dotted line) folds

Synth–mat dataset

The Synth–mat dataset is a 2 dimensional mixture of 4 normal distributions and has been presented in Figure 4.5(b). Both classes have bimodal distribution – there are two Gaussians in each of them. The mean value of correntropy between all 8 folds and the original dataset is 0.75 for DPS and 0.66 for CV averaged over 10 runs ($\sigma = 0.12$). Since the scatter plots of all DPS and CV folds were already presented for the Cone–torus dataset and not much changes here, only the decision boundaries of a classifier trained as previously have been given in Figure 4.9. Once again the boundaries appear stable for DPS and differ a lot from one CV fold to another.

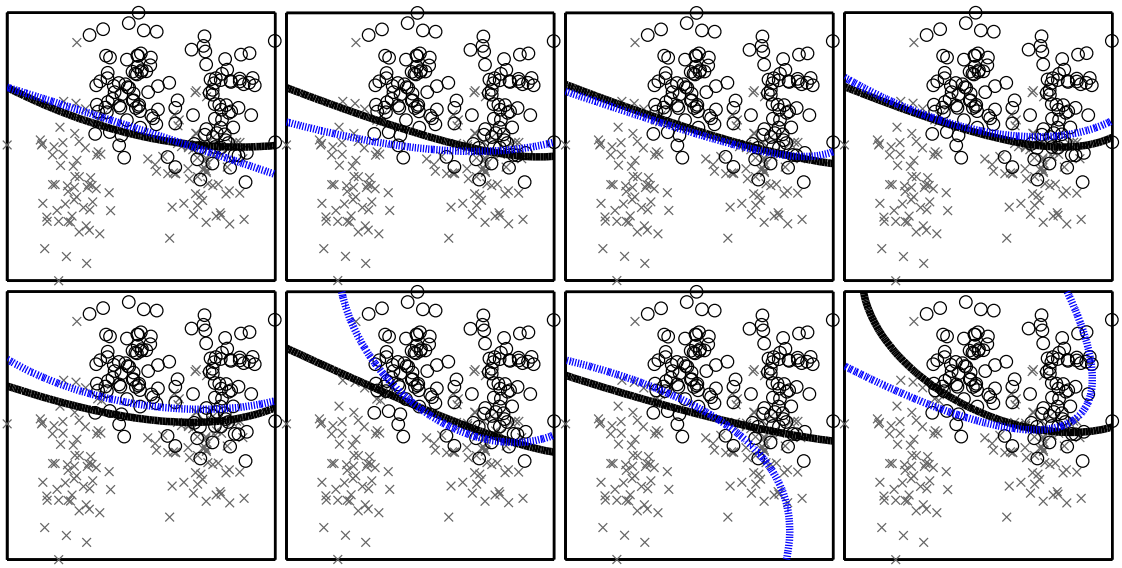


Figure 4.9: Synth–mat – decision boundaries for *qdc* trained on DPS–S (solid line) and CV (dotted line) folds

4.5.2 Benchmark datasets

Correntropy

Figure 4.10 presents the values of averaged correntropy between the original dataset and 8 folds generated using DPS and CV, for all 27 datasets used in the experiment. Note, that although the correntropy has been normalised to the $[0, +1]$ range, according to the earlier argument the values represent an ordinal scale. Also, the Gaussian kernel width used for each dataset has been chosen to maximise the correlation between bias and correntropy.

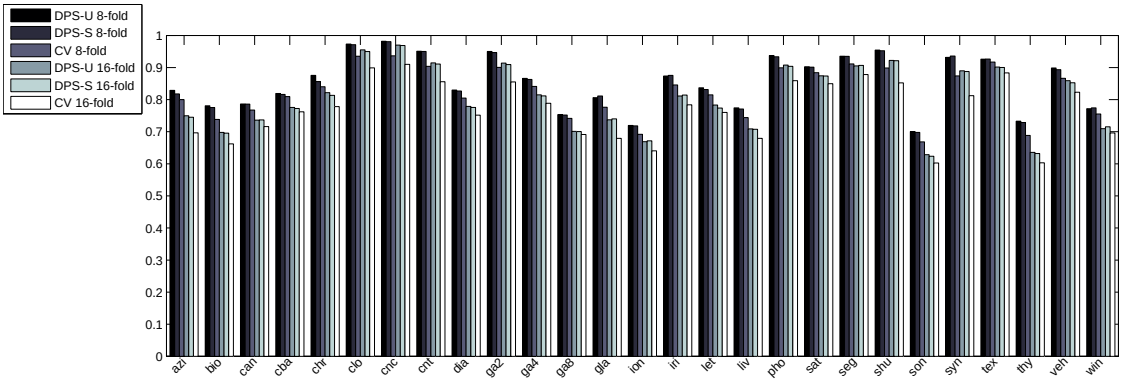


Figure 4.10: Mean correntropy between each fold and the original dataset

The correntropy between the DPS folds and the original dataset is always higher than in the case of the CV folds, regardless of the number of folds (8 or 16). This is not surprising since the DPS splits have been obtained by maximisation of correntropy. The picture is very similar for the between–fold correntropy depicted in Figure 4.11, where DPS is again an unquestionable leader.

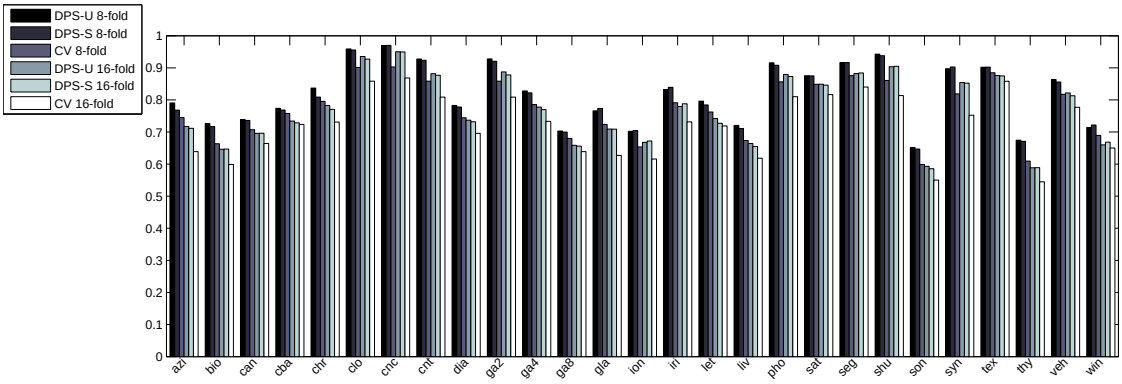


Figure 4.11: Mean between–fold correntropy

Bias

The mean absolute bias for both DPS and CV can be seen in Figures 4.12 and 4.13. The DPS approach has a bias comparable to the mean CV result, with slight advantage of the latter for roughly half of the datasets. Note however, that the DPS estimates are never as biased as

the worst–case CV scenario, yet the result has been achieved with approximately 10 times less computations.

A summary of the results can be found in Table 4.1, where a mean value and standard deviation of bias (and variance) across all datasets and classifiers for each error estimation method have been given. Both DPS–U and DPS–S have on average the same bias with a tiny difference in its standard deviation. DPS–SU on the other hand comes very close to the repeated cross–validation, which is a result of combining both supervised and unsupervised methods. Note, that this combination does not require additional computations in order to obtain the splits, as all pairwise within–class distances form a subset of all pairwise distances for the whole dataset, which are calculated anyway by the unsupervised DPS. All DPS approaches also have mean bias and standard deviation lower than the worst–case CV scenario.

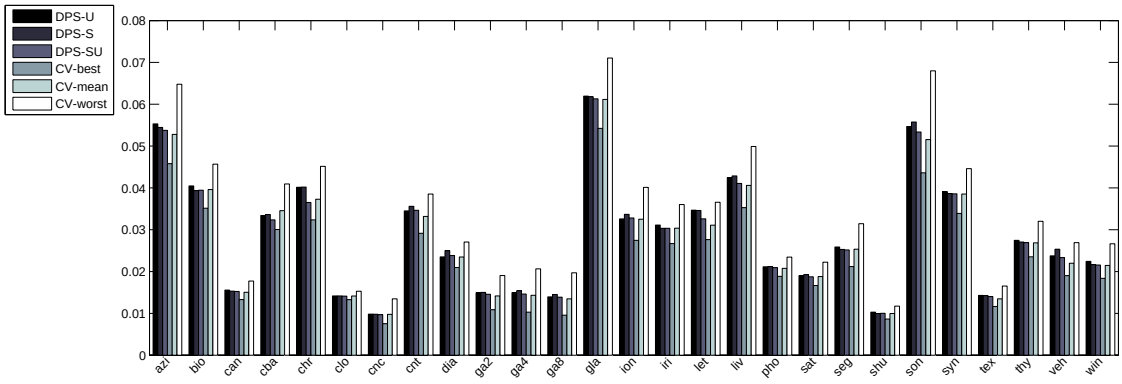


Figure 4.12: Mean absolute bias (averaged over all classifiers)

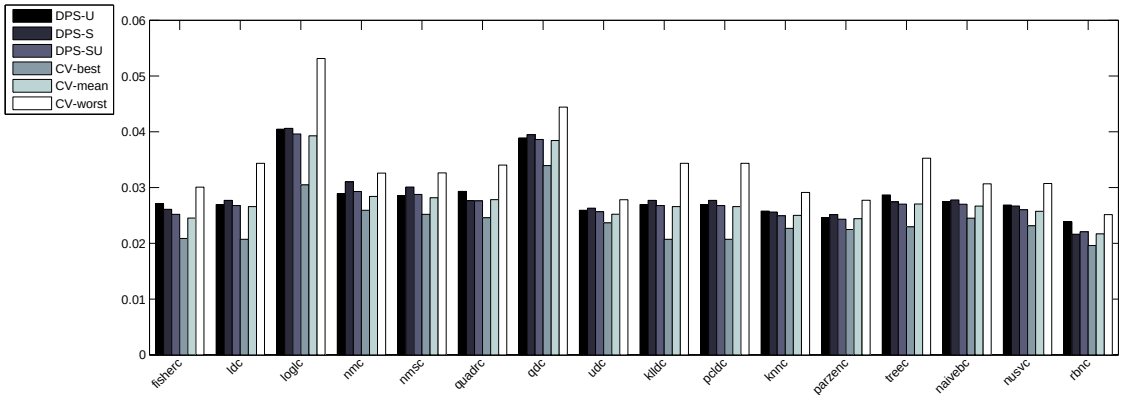


Figure 4.13: Mean absolute bias (averaged over all datasets)

Variance

The variance of error estimates can be seen in Figure 4.14 (averaged over all classifiers) and Figure 4.15 (averaged over all datasets). Out of all three DPS approaches, once again DPS–SU demonstrates the best performance with average variance lower by 0.013 than the best–case CV scenario (Table 4.1), while DPS–S performs at the level of best–case CV and DPS–U still outperforms 10 times repeated cross–validation. Note, that in terms of variance, DPS–S

Table 4.1: Bias and variance summary

	DPS–U	**DPS–S**	**DPS–SU**	**CV best**[a]	**CV mean**	**CV worst**
BIAS–mean	0.0281	0.0281	0.0275	0.0239	0.0273	0.0326
BIAS–stdev	0.0201	0.0202	0.0198	0.0164	0.0194	0.0242
VARIANCE–mean	0.0597	0.0504	0.0394	0.0524	0.0627	0.0743
VARIANCE–stdev	0.0332	0.0327	0.0229	0.0304	0.0345	0.0397

[a] 'CV best' denotes the best cross–validation run out of 10 for each dataset/classifier pair in terms of lowest bias/variance. For CV the division of data which produced the lowest bias did not in general produce the lowest variance. Similar remarks apply to 'CV worst'.

outperforms DPS–U and is additionally computationally cheaper (see Section 4.4.2). As a result good error estimation can be achieved with roughly 10% of computations required by 10 times repeated CV. For best results however one should resort to DPS–SU, which seems to stabilise the error estimates, requiring about 20% of the computations of 10 times repeated CV.

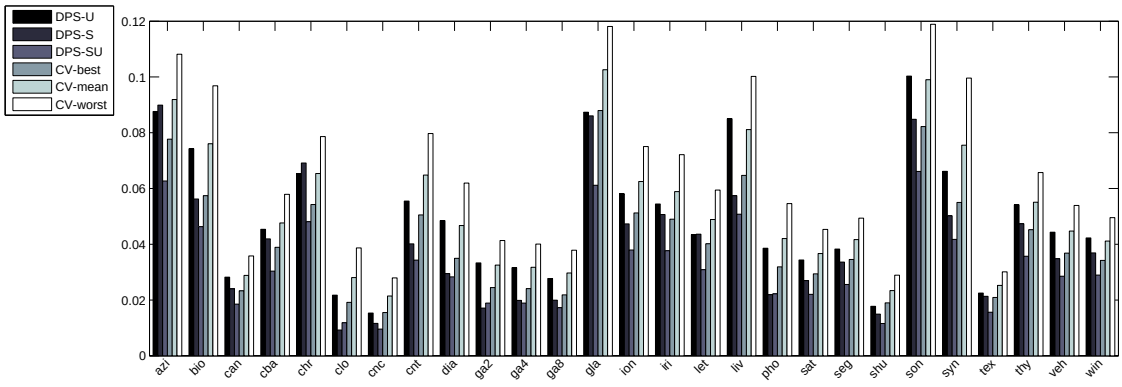


Figure 4.14: Standard deviation of error estimate (averaged over all classifiers)

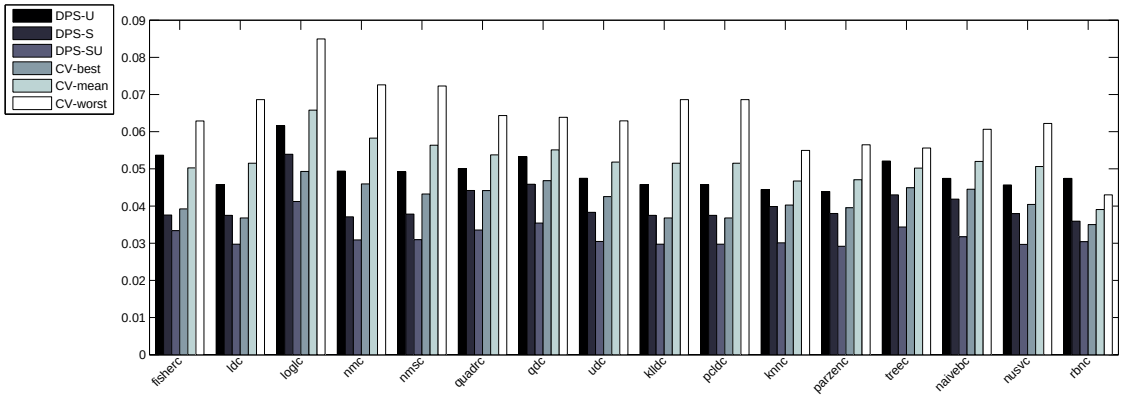


Figure 4.15: Standard deviation of error estimate (averaged over all datasets)

Classifier selection

Selection of a single best model from a group of available models is an important problem in machine learning. A typical selection criterion is the generalisation error estimated using cross–validation. The ranking of top 3 classifiers according to both CV and DPS for all datasets is given in Table 4.2. Note, that the overall ranking for all datasets is exactly the same for both error estimators, and reflects the ranks based on the true generalisation error. The differences are however apparent when the results for each dataset are examined separately.

Table 4.2: Ranking of top 3 classifiers

dataset	true	DPS–U	DPS–S	DPS–SU	CV
azi	11 14 12	11 12 2	11 12 14	11 12 2	11 12 14
bio	6 7 8	6 7 8	6 7 8	6 7 8	6 7 8
can	12 2 9	12 2 9	15 12 2	15 12 2	12 15 2
cba	12 2 9	12 2 9	12 2 9	12 2 9	12 2 9
chr	14 8 11	14 8 11	14 8 2	14 8 11	14 8 11
clo	15 1 2	1 2 3	15 1 2	15 1 2	15 1 2
cnc	1 15 3	1 15 2	1 15 5	1 15 5	1 15 5
cnt	16 12 13	16 12 7	12 16 13	16 12 13	16 12 13
dia	1 3 2	2 9 10	1 11 2	1 2 9	1 3 2
ga2	1 2 3	15 4 2	2 3 9	2 3 9	3 1 2
ga4	1 2 3	1 4 2	15 5 2	15 1 2	15 4 3
ga8	16 3 1	16 1 5	16 5 4	16 5 1	16 15 5
gla	3 15 2	2 9 10	2 9 10	2 9 10	2 9 10
ion	14 11 7	14 7 11	14 13 7	14 13 7	14 7 11
iri	2 9 10	2 9 10	2 7 9	2 9 10	2 9 10
let	11 12 7	12 11 2	12 11 2	12 11 2	12 11 2
liv	1 3 2	1 3 2	11 1 3	1 3 11	1 3 11
pho	11 16 12	16 12 11	11 12 16	11 16 12	11 12 16
sat	11 12 2	11 12 14	11 12 2	11 12 2	11 12 7
seg	11 3 2	3 11 2	11 2 9	11 3 2	3 11 2
shu	13 1 16	13 16 1	13 1 16	13 16 1	13 1 16
son	12 11 15	12 11 14	12 11 14	12 11 14	12 11 2
syn	14 12 16	14 12 16	12 14 16	14 12 16	12 14 16
tex	2 9 10	2 9 10	2 9 10	2 9 10	2 9 10
thy	8 15 7	15 6 7	7 8 11	15 7 8	15 8 11
veh	7 6 3	6 7 2	7 6 2	7 6 2	7 6 2
win	7 6 2	6 1 4	4 6 7	6 4 7	6 7 4
overall	2 9 10	2 9 10	2 9 10	2 9 10	2 9 10
in top 1	27/27	17/27	17/27	20/27	19/27
in top 2	27/27	20/27	23/27	23/27	25/27
in top 3	27/27	21/27	24/27	25/27	25/27

The last three rows in the table denote the number of datasets out of 27, for which the true top classifier was included in top 1, top 2 and top 3 classifiers according to each error estimation

method. For CV, the best classifier has been correctly identified 19 times and has been included in the top 2 and top 3 classifiers 25 times. For the best DPS approach (DPS–SU) the numbers are very similar – 20, 23 and 25.

The correlation coefficients for different error estimates and true generalisation error are given in Table 4.3. As shown, all tested error estimators are strongly correlated with the true error, and the correlation coefficient is never lower than 0.959.

Table 4.3: Correlation between true generalisation error and error estimates

correlation	**DPS–U**	**DPS–S**	**DPS–SU**	**CV**
per dataset (8 folds)	0.9594	0.9657	0.9676	0.9710
per classifier (8 folds)	0.9967	0.9975	0.9976	0.9973
per dataset (16 folds)	0.9640	0.9646	0.9671	0.9695
per classifier (16 folds)	0.9964	0.9969	0.9969	0.9969

Correlation between correntropy and bias

The ability to estimate the generalisation error using a single DPS fold only would allow to reduce the computational cost of the estimation procedure by another order of magnitude, when compared to 10 times repeated CV. Figure 4.16 depicts the bias of the estimate calculated using a single DPS fold, which has been chosen on the basis of the lowest bias itself ('DPS–best'). Although in practice this kind of selection procedure is infeasible, it shows that the method has some potential as for most datasets the bias is comparable with the one obtained using 10 times repeated cross–validation or even the best–case CV scenario. The problem however is how to choose the appropriate DPS fold. The value of correntropy seems to be an obvious choice. Note however, that there is no principled way of selecting the width σ of the Gaussian kernel for estimation of correntropy and the estimated value can vary greatly depending on the choice of σ. It has therefore been decided to check the correlation between bias and the values of correntropy. The experiment was performed for 8 and 16 DPS–S folds and the results can be seen in Figure 4.17. Note, that for the sake of calculating the correntropy, σ was chosen using an exhaustive search in order to optimise the correlation. In other words, the results given in Figure 4.17 represent the best–case scenario, for the most optimal kernel width, which in practice is not known a priori. As it can be seen, the correlation varies from about -0.1 to -0.6 depending on the dataset. The bias of an estimate obtained using a single DPS fold chosen on the basis of highest correntropy is always higher even in comparison to the worst–case CV scenario bias ('DPS–optim' in Figure 4.16). The estimate of correntropy is only slightly to moderately correlated with the bias of the error estimate, even for an optimal choice of Gaussian kernel width. As a result it cannot be used to select a single best fold which would minimise the bias, although some other divergence measures might be appropriate for this task. This issue is further investigated in Chapter 5.

Combining classifiers

In this experiment a simple ensemble model based on the majority voting rule was built. It is believed, that the classifiers used in a combination should be diverse, which enforces

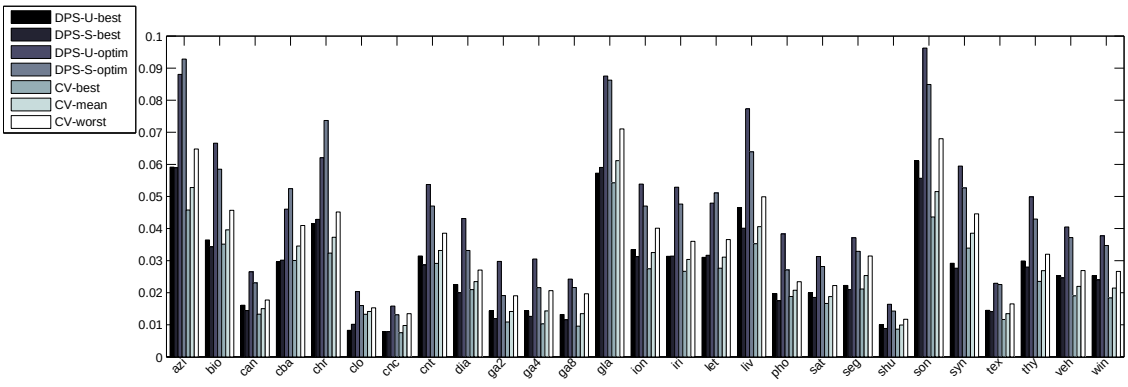


Figure 4.16: Bias of DPS error estimate calculated using a single fold (averaged over all classifiers)

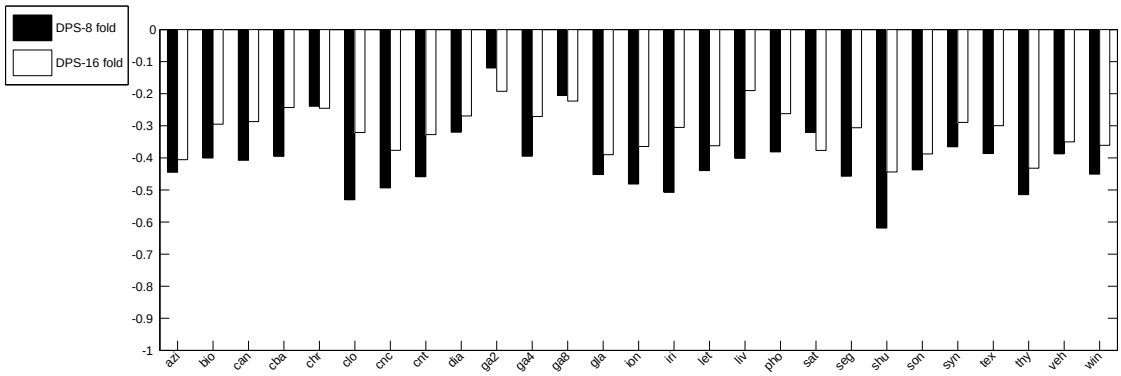


Figure 4.17: Correlation between bias and correntropy

complementarity of the ensemble members [140]. One way to enforce this diversity is cross–training, a technique based on cross–validation, which combines all models obtained during a single or repeated CV run. For this experiment two synthetic datasets described in Section 4.5.1 have been used. Both datasets were split into 8 folds using DPS–S and CV and then for each classifier from the previous experiment an ensemble model was built by combining 8 models trained on all but one fold in turn and using the majority voting rule. For CV this procedure was repeated 10 times. Each combination was then tested on independent test set. In order to monitor performance of the combinations, a single control model trained using all 8 folds was also used.

The results have been depicted in Figures 4.18 and 4.19. In most cases, combinations based on DPS folds do not improve on the performance of a single control model. This was expected, as for each classifier all 8 ensemble members should be very similar, since they were all trained using representative subsets of data. For the combinations based on CV, some improvement can be observed even in the worst case scenario.

In order to illustrate this issue, discrete error distribution plots showing the probability of various numbers of ensemble members being in error at the same time have been given in Figures 4.20 and 4.21. The classifiers used to produce these plots (*qdc* and *treec*) have been chosen primarily for illustrative purposes. The area of the gray–shaded region in each figure represents the error of the combined model. Usually the number of models in majority voting is chosen to be odd, so that there are no ties. In the presented case there are however 8 models,

so the ties were resolved randomly. Note, that for DPS most of the mass is concentrated on the sides of the plots, meaning that the classification decisions are taken unanimously in most cases, proving that the classifiers are indeed very similar.

In case of CV the situation is different. In Figure 4.20 for example some mass is scattered all over the plot, meaning that there are situations when the classifiers tend to disagree and hence

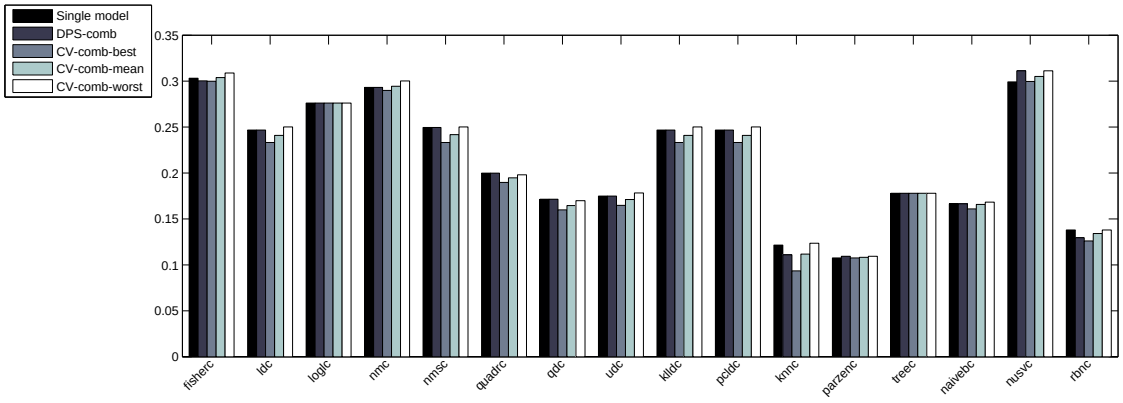


Figure 4.18: Single model v. combination errors for Cone–torus dataset

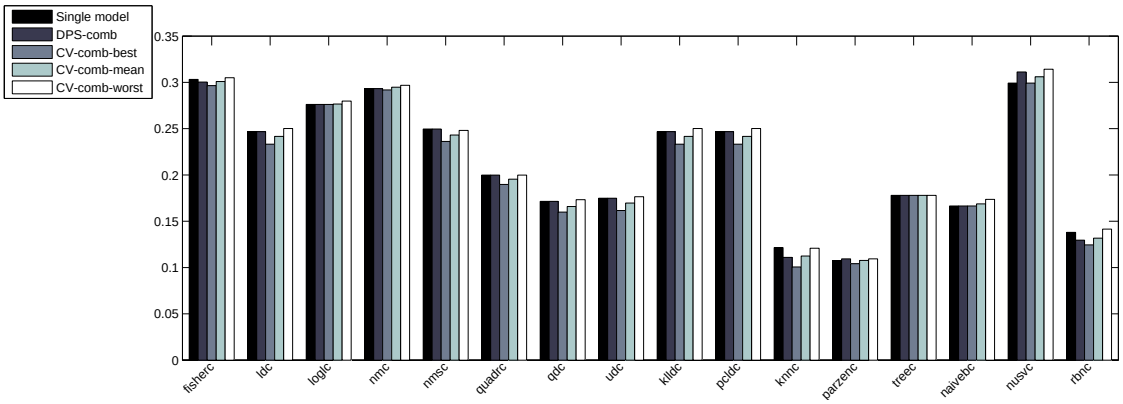
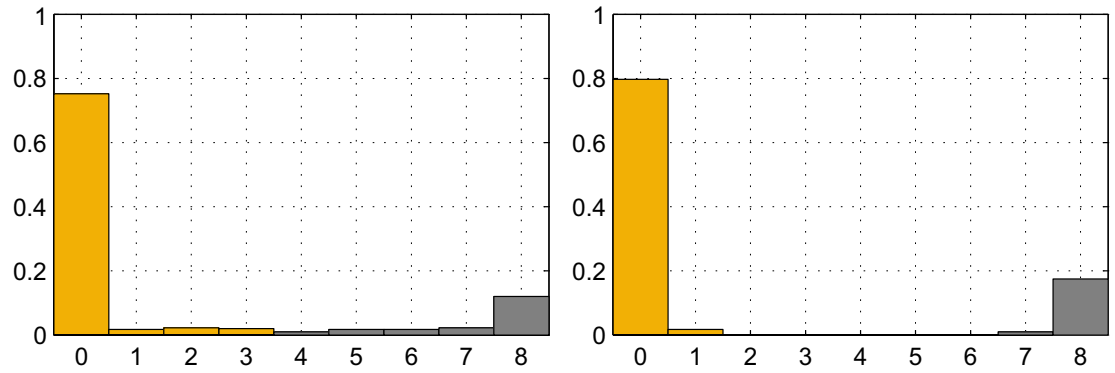
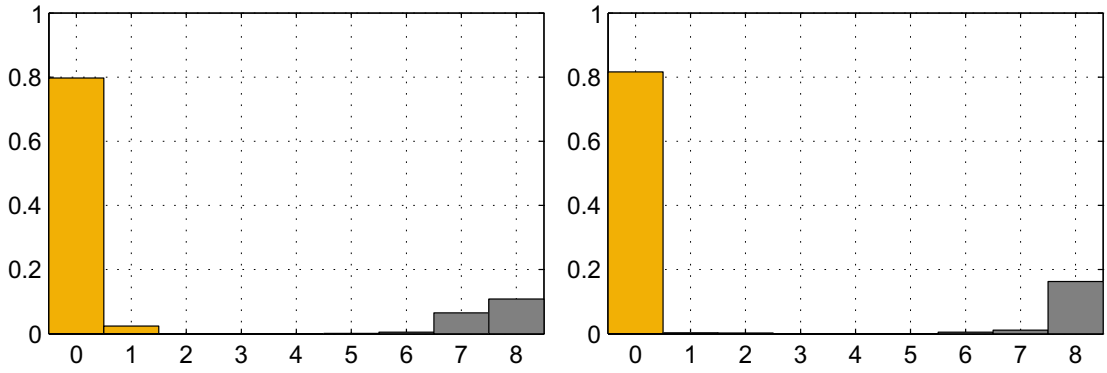


Figure 4.19: Single model v. combination errors for Synth–mat dataset



(a) CV–best (0.1616) (b) DPS (0.1715)

Figure 4.20: Discrete Error Distributions for Cone–torus dataset and *qdc* (error rates given in brackets)



(a) CV–best (0.1790) (b) DPS (0.1790)

Figure 4.21: Discrete Error Distributions for Synth–mat dataset and *treec* (error rates given in brackets)

demonstrate complementarity. The stochastic nature of CV in this case has a very positive effect on the performance by introducing diversity to the ensemble. This result also confirms, that if the goal is to select a single best model, it is much safer to use DPS as this minimises the risk of choosing a bad model due to the stability of decision boundaries as discussed before. From the point of view of diversity required for ensemble models however, this feature of DPS becomes a disadvantage and it's usually much better to use a stochastic method instead.

4.6 Discussion

The presented Density Preserving Sampling procedure is a very attractive alternative for the commonly used cross–validation technique for a number of reasons.

For the purpose of the generalisation error estimation, k–fold cross–validation is without a doubt the most widely and commonly used technique, due to its universal character, simplicity and effectiveness. Its stochastic nature however requires the estimation to be repeated multiple times for different random divisions of the data, in order to circumvent the risk of obtaining the best/worst–case scenario estimate, which as demonstrated in this work can be highly biased and can have a large variance. The need for running the procedure multiple times makes it computationally expensive, forcing the researchers to seek compromise elsewhere, for example by not calculating the full gradient during optimisation or taking other shortcuts, which negatively influence the solution of the problem. The DPS procedure proposed in this thesis is however deterministic. It thus does not need to be repeated in order to improve the quality of the error estimate, at the same time producing results comparable to repeated cross–validation when it comes to bias, and superior to CV in terms of the variance of obtained estimates. Yet it all happens at 5–10 times lower computational cost.

Another related application area of CV is parameter estimation. Since for some models the objective function is not differentiable with respect to all its parameters, the optimisation procedure must resort to a search in the parameter space. One example of such situation is the kNN classifier, for which the number of nearest neighbours *k* is usually being set by testing a number of possible values using cross–validation. In such case, as the search itself might be very costly depending on the dimensionality of the search space, the cross–validation is usually not being repeated in order to save computations. As before, due to the non–deterministic nature of CV, this can lead to suboptimal decisions based on highly biased performance estimates

(worst–case scenario). Note, that it also applies to other algorithms requiring calculation of performance estimates repeated many times like e.g. feature selection. The benefit of using DPS rather than CV in these scenarios can be tremendous.

In case of some machine learning methods it is a common practice to cross–train multiple models and select the best performing one. The cross–training procedure is analogous to cross–validation, with the difference that the obtained models instead of being discarded, are considered as candidates for a final solution. This applies especially to models like decision trees, which cannot be retrained using the full dataset due to their instability. The danger here is the combination of a relatively unstable error estimation procedure (see plots of the decision boundaries in Figures 4.8 and 4.9) with an unstable learning method, which in an unfavorable case may lead to selection of one of the worst models rather than the best. On the other hand, models trained using various DPS splits will likely be much more similar to each other, as shown in Section 4.5.2, minimising the risk and cost of incorrect choice.

Yet another possible application of random sampling procedures is early stopping, a technique widely used in training of universal approximators (e.g. neural networks) to prevent overfitting. In this approach a randomly selected subset of the data is used for continuous monitoring of model performance during training, in order to stop it when the validation error starts to increase, which is a sign of overfitting. The risk of using unrepresentative validation set is obvious in this case. Although the behaviour of DPS in conjunction with early stopping has not been addressed here, it forms an interesting and promising research direction.

4.7 Concluding remarks

The proposed correntropy–based DPS procedure is an interesting alternative for widely used cross–validation technique in many applications. Unlike CV, DPS is a deterministic method, which eliminates the need for multiple repetitions of the sampling procedure to obtain reliable results, considerably reducing the computational burden.

The main property of the proposed method is that it aims to produce representative splits, which has many implications outlined in the previous section. The experiments conducted using a diverse set of publicly available benchmark datasets have revealed that:

- For generalisation error estimation, DPS is slightly more biased than 10 times repeated cross–validation but it has low variance, often lower than the best–case CV scenario. The DPS bias in all cases also appears much lower than in the worst–case CV scenario.
- The decision boundaries of a classifier trained using DPS folds seem much more stable than in the case of a single cross–validation folds, which is the result of representativeness of the subsets generated by DPS. The stability of models trained on various DPS divisions of the dataset has been confirmed in experiments involving ensemble models.
- For model ranking and selection, DPS is at least as good as 10 times repeated cross–validation, at much lower computational cost.

The DPS procedures described in this thesis have been formally externally approved for the inclusion in the PRTools toolbox [42].

Chapter 5

PDF divergence estimators and their applicability to representative data sampling

5.1 Introduction

The Density Preserving Sampling procedure derived in Chapter 4 proved to be comparable to repeated cross–validation for the purpose of estimation of model generalisation ability, while eliminating the need for repetitions in the estimation process. It has also been demonstrated that it is possible to select only a single DPS fold and use it as validation data to obtain an error estimate with accuracy comparable with 10 times repeated cross–validation, effectively reducing the computational requirements by another order of magnitude. The problem of selecting the right DPS fold however, still remains unsolved. The correntropy [102], which is the objective function used in DPS optimisation, is only moderately correlated with bias of the error estimate. Moreover, the correlation is only apparent for a single, carefully chosen value of the kernel smoothing parameter, but there is no principled way to discover this value. The idea of further reduction of computational cost of generalisation error estimation nevertheless still remains very attractive.

In this chapter the possibilities of selecting a representative subset of data from a larger dataset are further investigated. Unfortunately, there is no universal and measurable notion of representativeness. A standard definition of a representative sample that can be found in any statistical textbook (e.g. [37]) states that it should have the same properties as the population from which it has been drawn. The question 'which properties' however remains open and it is fair to assume, that the answer differs from one application to another. In the case pursued in this chapter the application can be stated as accurate estimation of the generalisation performance of a predictive model. For this purpose easily calculable measure of representativeness, based on the probability density functions as the most universal characteristic of data, is required. A number of most popular divergence measures that can be found in the literature is thus examined, in order to investigate their usability for a given goal.

There are however some challenges here. First of all, in most real–world applications the PDFs have unknown, non–parametric forms and as a result need to be approximated somehow. The two best known and most commonly used methods of doing this are the Parzen window [121] and k–Nearest Neighbour (kNN) [40] density estimation. The problem is, that if the estimates of the PDFs are poor, it's hard to expect the value of a divergence measure calculated using these

estimates to be reliable. The PDF estimates can be inaccurate for many reasons like incorrect choice of the parameters of density estimation methods, not enough data used for the estimation, or non–representativeness of the data. Moreover, as the divergence measures in use are defined as integrals over various functions of the PDFs, in all but the simplest cases there are no closed–form formulas for their calculation. The only way is to resort to some estimation procedure, which can make the discussed problem even worse. After all, if one discovers that there is no correlation between the estimate of the divergence measure and bias of the error estimate, is it because the divergence measure is not suitable for this task, or because its estimate is poor? Or maybe the poor estimate of the underlying PDF is to blame here?

5.2 Estimation of the probability density functions

Before presenting various divergence measures and their estimators, estimation of the PDFs directly from data is first discussed. This issue is fundamental in the context of divergence estimation, for at least two reasons: (1) the divergence is measured between two or more PDFs, so some expressions for the PDFs are obviously needed and (2) the PDFs very rarely have known parametric forms. Two most popular non–parametric density estimation methods are presented below.

5.2.1 Parzen window method

The Parzen window method [121] is the most commonly used non–parametric PDF estimation procedure. The estimate $\hat{f}(\boldsymbol{x})$ of the unknown density $f(\boldsymbol{x})$ can be obtained by using:

$$\hat{f}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{V_N}\varphi\left(\frac{\boldsymbol{x}-\boldsymbol{x}_i}{\sigma_N}\right) \tag{5.1}$$

where N is the dataset size, V_N stands for window volume, φ is a window function and σ_N is the smoothing parameter also known as window width or bandwidth [40]. The window function is often chosen to be a Gaussian due to its analytical properties. Using a Gaussian function, the two unknown PDFs $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$, which will be extensively used in further sections, can be approximated in the same way as in Eq. 4.5:

$$\hat{p}(\boldsymbol{x}) = \frac{1}{N_p}\sum_{i=1}^{N_p} G(\boldsymbol{x}-\boldsymbol{x}_{pi}, \sigma_p^2\boldsymbol{I}) \tag{5.2}$$

$$\hat{q}(\boldsymbol{x}) = \frac{1}{N_q}\sum_{i=1}^{N_q} G(\boldsymbol{x}-\boldsymbol{x}_{qi}, \sigma_q^2\boldsymbol{I}) \tag{5.3}$$

where N_p is the number of $d-$dimensional points drawn i.i.d. according to $p(\boldsymbol{x})$: $\mathcal{X}_p = \{\boldsymbol{x}_{p1}, \boldsymbol{x}_{p2}, \ldots, \boldsymbol{x}_{pN_p}\}$, N_q is the number of points drawn i.i.d. according to $q(\boldsymbol{x})$: $\mathcal{X}_q = \{\boldsymbol{x}_{q1}, \boldsymbol{x}_{q2}, \ldots, \boldsymbol{x}_{qN_q}\}$ and $G(\boldsymbol{x}-\boldsymbol{x}_i, \sigma^2\boldsymbol{I})$ is a spherical Gaussian PDF given by:

$$G(\boldsymbol{x}-\boldsymbol{x}_i, \sigma^2\boldsymbol{I}) = \frac{1}{(2\pi\sigma^2)^{d/2}}\ \exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{x}_i)^T(\boldsymbol{x}-\boldsymbol{x}_i)}{2\sigma^2}\right) \tag{5.4}$$

The Gaussian PDF of Eq. 5.4 corresponds to the window function with absorbed normalising constant V_N. Note, that the Gaussian is spherical and is thus characterised by a single parameter σ, rather than by the whole covariance matrix. Although intuitively it limits flexibility, the estimators of Eqs. 5.2 and 5.3 converge to the true densities when $N \to \infty$, if at the same time the window bandwidth tends to 0 at a certain rate. When dealing with two subsamples of the same dataset, it can be assumed that $\sigma_p = \sigma_q = \sigma$, effectively eliminating one of the parameters.

Given a set of data, the PDF estimation now comes down to a single parameter σ, hence its value should be selected very carefully. If this value is chosen too big, the density estimate will be oversmoothed and the fine details of the PDF will be lost. If σ is chosen too small, the estimate will be very spiky with large regions of values near 0. Thus there has been a substantial amount of research focused on automatic selection of the bandwidth parameter from the data (a review can be found in [174]). Three different bandwidth selection methods have been used in this study:

- **Pseudo likelihood cross–validation** [41], which selects the bandwidth σ to maximise a pseudo–likelihood function of the density estimate using leave–one–out approximation to avoid a trivial maximum at $\sigma = 0$. Interestingly, the pseudo–likelihood method minimises the Kullback–Leibler divergence between the true density and the estimated density, but it tends to produce inconsistent estimates for heavy–tailed PDFs [174].
- **'Rule of Thumb' (RoT)** method [158], which minimises the Asymptotic Mean Integrated Squared Error (AMISE) between the true distribution and its estimate. Calculation of bandwidth minimising the AMISE criterion requires estimation of integral of squared second derivative of the unknown true PDF [174], which is difficult. To overcome this issue, the RoT method replaces the unknown value with an estimate calculated with reference to a normal distribution. This makes the method computationally inexpensive at the risk of producing poor estimates for distributions which depart from Gaussian.
- **'Solve–the–equation plug–in' method** [157], which also minimises AMISE between the true distribution and its estimate, but without assuming any parametric form of the former. This method is currently considered as state–of–the–art [88], although it has a computational complexity which is quadratic in the dataset size. In the experiments a fast approximate bandwidth selection algorithm of [130], which scales linearly in the size of data, has been used.

5.2.2 k–Nearest Neighbour method

The second well known probability density estimator is the k–Nearest Neighbour method, according to which densities $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$ can be approximated as [40, 124]:

$$\tilde{p}(\boldsymbol{x}) = \frac{k\Gamma(d/2+1)}{N_p \pi^{d/2} r_k(\boldsymbol{x})^d} \tag{5.5}$$

$$\tilde{q}(\boldsymbol{x}) = \frac{k\Gamma(d/2+1)}{N_q \pi^{d/2} s_k(\boldsymbol{x})^d} \tag{5.6}$$

where k is the nearest neighbour count, $\pi^{d/2}/\Gamma(d/2+1)$ is the volume of a unit–ball and $r_k(\boldsymbol{x})$, $s_k(\boldsymbol{x})$ are the Euclidean distances from $\boldsymbol{x}$ to its k^{th} nearest neighbour in $\mathcal{X}_p$ and $\mathcal{X}_q$ respectively. Note, that if $\boldsymbol{x} \in \mathcal{X}_p$ then the influence of $\boldsymbol{x}$ on the density estimate should be eliminated, thus N_p in Eq. 5.5 becomes $N_p - 1$ and $r_k(\boldsymbol{x})$ denotes the distance to the k^{th} nearest neighbour in $\mathcal{X}_p \backslash\, x$ rather than in $\mathcal{X}_p$. A similar remark applies to the situation when $\boldsymbol{x} \in \mathcal{X}_q$.

5.3 Probability density function divergence measures

There are many different divergence measures that one can find in the literature [26]. Perhaps the most prominent of them is the family of Chernoff's α–divergences [95], which includes such measures as the Kullback–Leibler divergence [96] or squared Hellinger's distance [100] as its special cases. Although most of these measures have strong theoretical foundations, there are no closed–form solutions to calculate them, apart from the cases when the probability density functions are some simple parametric models like e.g. Gaussians. For this reason, one is forced to resort to some kind of estimators of the divergences. The estimators can be obtained in various ways and one of them is to estimate the unknown probability density functions first and then substitute them into the formulas for divergences. This is the approach taken in this study. Note, that the literature on PDF divergence estimation is somewhat inconsistent, hence in the following sections an attempt has been made to gather all the relevant concepts and present them in a unified way, filling some of the existing gaps. As a result, most of the formulas that can be found in the Sections 5.3 and 5.4 have been derived or transformed for the purpose of this study.

Throughout the following sections the sample mean is extensively used as the estimator of expected value. By the Law of Large Numbers sample mean converges to the expected value with probability 1, as the sample size tends to infinity. The expected value of an arbitrary function $q(\boldsymbol{x})$, w.r.t. the PDF $p(\boldsymbol{x})$ is:

$$E_{p(x)}\left[q(\boldsymbol{x})\right] = \int q(\boldsymbol{x})\, p(\boldsymbol{x})\, d\boldsymbol{x} \tag{5.7}$$

and can be approximated by:

$$\hat{E}_{p(x)}\left[q(\boldsymbol{x})\right] = \frac{1}{N}\sum_{i=1}^{N} q(\boldsymbol{x}_i) \approx E_{p(x)}\left[q(\boldsymbol{x})\right] \tag{5.8}$$

5.3.1 Kullback–Leibler divergence

The Kullback–Leibler divergence (D_{KL}), also known as information divergence or relative entropy, is probably the most widely used measure of similarity between two probability density functions [96]. The measure has been used in a wide variety of applications, like data condensation [53], Blind Source Separation via Independent Component Analysis [22, 23], classification [68, 115], or image processing [14, 112] to name a few. The Kullback–Leibler divergence between the joint PDF and a product of marginal PDFs is equal to the mutual information between the two random variables, which is one of the most important concepts of information theory [106].

The Kullback–Leibler divergence is non–symmetric and can be interpreted as the number of additional bits (if base 2 logarithm is used) needed to encode instances from the true distribution $p(\boldsymbol{x})$ using the code based on some other distribution $q(\boldsymbol{x})$ [96]. The measure is thus directly related to Shannon's information theory. Kullback–Leibler divergence for two continuous random variables, which are of interest here, is given by:

$$D_{KL}(p,q) = \int p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\, d\boldsymbol{x} \tag{5.9}$$

The natural logarithms are used throughout this chapter unless otherwise stated. From the above definition it's easy to see, that D_{KL} is only defined if $q(\boldsymbol{x}) > 0$ for every $\boldsymbol{x}$. Using Eqs. 5.7

and 5.8 one can also write:

$$D_{KL}(p,q) = E_{p(x)}\left[\log\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\right] \approx \frac{1}{N_p}\sum_{i=1}^{N_p}\log\frac{p(\boldsymbol{x}_{pi})}{q(\boldsymbol{x}_{pi})} \tag{5.10}$$

Finally, by substituting Eqs. 5.2 and 5.3 into Eq. 5.10 and rearranging, the formula for the estimator $\hat{D}_{KL}(p,q)$ becomes:

$$\begin{aligned}\hat{D}_{KL}(p,q) &= \frac{1}{N_p}\sum_{i=1}^{N_p}\log\frac{\hat{p}(\boldsymbol{x}_{pi})}{\hat{q}(\boldsymbol{x}_{pi})} = \frac{1}{N_p}\sum_{i=1}^{N_p}\left[\log\hat{p}(\boldsymbol{x}_{pi}) - \log\hat{q}(\boldsymbol{x}_{pi})\right] \qquad (5.11)\\ &= \frac{1}{N_p}\sum_{i=1}^{N_p}\left[\log\frac{1}{N_p}\sum_{j=1}^{N_p}G(\boldsymbol{x}_{pi}-\boldsymbol{x}_{pj},\sigma^2\boldsymbol{I}) - \log\frac{1}{N_q}\sum_{j=1}^{N_q}G(\boldsymbol{x}_{pi}-\boldsymbol{x}_{qj},\sigma^2\boldsymbol{I})\right]\end{aligned}$$

In the experiments in Section 5.4 another estimator of D_{KL} derived in [124, 179], based on the kNN density estimate rather than Parzen window, is also used:

$$\tilde{D}_{KL}(p,q) = \frac{d}{N_p}\sum_{i=1}^{N_p}\log\frac{s_k(\boldsymbol{x}_{pi})}{r_k(\boldsymbol{x}_{pi})} + \log\frac{N_q}{N_p-1} \tag{5.12}$$

where $r_k(\boldsymbol{x}_{pi})$ and $s_k(\boldsymbol{x}_{pi})$ are the Euclidean distances to the k^{th} nearest neighbor of $\boldsymbol{x}_{pi}$ in $\mathcal{X}_p \setminus \boldsymbol{x}_{pi}$ and $\mathcal{X}_q$ respectively.

For a special case, when both $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$ are Mixtures of Gaussians there exist other techniques for approximation of D_{KL}, which have been reviewed in [72].

5.3.2 Jeffrey's divergence

One of the inconveniences of the Kullback–Leibler divergence is the fact, that it is non–symmetric. Jeffrey's divergence (D_J) is a simple way of making D_{KL} symmetric and is given by the following formula [82]:

$$D_J(p,q) = \int (p(\boldsymbol{x}) - q(\boldsymbol{x}))(\log p(\boldsymbol{x}) - \log q(\boldsymbol{x}))\, d\boldsymbol{x} \tag{5.13}$$

Note, that after rearranging:

$$D_J(p,q) = \int p(\boldsymbol{x})\log\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\, d\boldsymbol{x} - \int q(\boldsymbol{x})\log\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\, d\boldsymbol{x} = D_{KL}(p,q) + D_{KL}(q,p) \tag{5.14}$$

which solves the non–symmetricity issue in a very simple and intuitive way. Note however, that there is another problem: D_J is not defined if either $p(\boldsymbol{x}) = 0$ or $q(\boldsymbol{x}) = 0$, which is in fact even more restrictive than in the case of D_{KL}.

Jeffrey's divergence has been used for example in [112] for classification of multimedia data with Support Vector Machines.

5.3.3 Jensen–Shannon divergence

The Jensen–Shannon divergence (D_{JS}) is a measure designed to address the weaknesses of Kullback–Leibler divergence. Namely, unlike the latter, D_{JS} is symmetric, always finite and semibounded [101]. Jensen–Shannon divergence is defined in terms of D_{KL} as:

$$D_{JS}(p,q) = \frac{1}{2}D_{KL}(p,m) + \frac{1}{2}D_{KL}(q,m) \tag{5.15}$$

where $m(\boldsymbol{x}) = \frac{1}{2}(p(\boldsymbol{x}) + q(\boldsymbol{x}))$. Unfortunately no estimator of D_{JS} was given in [101], but it can be approximated using the estimators of D_{KL} as:

$$\begin{aligned} \hat{D}_{JS}(p,q) &= \frac{1}{2}\hat{D}_{KL}(p,m) + \frac{1}{2}\hat{D}_{KL}(q,m) \\ &= \frac{1}{2N_p}\sum_{i=1}^{N_p}[\log\hat{p}(\boldsymbol{x}_{pi}) - \log\hat{m}(\boldsymbol{x}_{pi})] + \frac{1}{2N_q}\sum_{i=1}^{N_q}[\log\hat{q}(\boldsymbol{x}_{qi}) - \log\hat{m}(\boldsymbol{x}_{qi})] \end{aligned} \tag{5.16}$$

where

$$\hat{m}(\boldsymbol{x}) = \frac{1}{2}(\hat{p}(\boldsymbol{x}) + \hat{q}(\boldsymbol{x})) = \frac{1}{2N_p}\sum_{i=1}^{N_p} G(\boldsymbol{x} - \boldsymbol{x}_{pi}, \sigma_p^2\boldsymbol{I}) \; + \; \frac{1}{2N_q}\sum_{i=1}^{N_q} G(\boldsymbol{x} - \boldsymbol{x}_{qi}, \sigma_q^2\boldsymbol{I}) \tag{5.17}$$

Using the kNN density estimator, the divergence can also be estimated as:

$$\begin{aligned} \tilde{D}_{JS}(p,q) &= \frac{1}{2N_p}\sum_{i=1}^{N_p}\log\frac{2\,N_q s_k(\boldsymbol{x}_{pi})^d}{N_q s_k(\boldsymbol{x}_{pi})^d + (N_p-1)r_k(\boldsymbol{x}_{pi})^d} \\ &+ \frac{1}{2N_q}\sum_{i=1}^{N_q}\log\frac{2\,N_p r_k(\boldsymbol{x}_{qi})^d}{(N_q-1)s_k(\boldsymbol{x}_{qi})^d + N_p r_k(\boldsymbol{x}_{qi})^d} \end{aligned} \tag{5.18}$$

Some of the applications of the Jensen–Shannon divergence include feature clustering for text classification [35] and outlier detection in sensor data [159].

5.3.4 Cauchy–Schwarz divergence

The Cauchy–Schwarz divergence D_{CS} is a symmetric measure obeying $0 \leq D_{CS} \leq \infty$, with the minimum obtained for $p(\boldsymbol{x}) = q(\boldsymbol{x})$ [129]. The measure has been inspired by the Cauchy–Schwarz inequality. It was derived as a part of the Information Theoretic Learning framework [125, 127] and its theoretical properties have been further investigated in [85]. The Cauchy–Schwarz divergence is given by the following formula:

$$D_{CS}(p,q) = -\log\frac{\int p(\boldsymbol{x})\,q(\boldsymbol{x})\,d\boldsymbol{x}}{\sqrt{\int p^2(\boldsymbol{x})\,d\boldsymbol{x}\int q^2(\boldsymbol{x})\,d\boldsymbol{x}}} \tag{5.19}$$

If the Parzen window method with Gaussian kernels is used for PDF estimation, substituting Eqs. 5.2 and 5.3 into each integral of Eq. 5.19 in turn and rearranging yields:

$$\int p(\boldsymbol{x})\, q(\boldsymbol{x})\, d\boldsymbol{x} = \frac{1}{N_p N_q} \sum_{i=1}^{N_p} \sum_{j=1}^{N_q} \int G(\boldsymbol{x} - \boldsymbol{x}_{pi}, \sigma_p^2 \boldsymbol{I})\; G(\boldsymbol{x} - \boldsymbol{x}_{qj}, \sigma_q^2 \boldsymbol{I})\, d\boldsymbol{x} \tag{5.20}$$

$$\int p^2(\boldsymbol{x})\, d\boldsymbol{x} = \frac{1}{N_p^2} \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} \int G(\boldsymbol{x} - \boldsymbol{x}_{pi}, \sigma_p^2 \boldsymbol{I})\; G(\boldsymbol{x} - \boldsymbol{x}_{pj}, \sigma_p^2 \boldsymbol{I})\, d\boldsymbol{x} \tag{5.21}$$

$$\int q^2(\boldsymbol{x})\, d\boldsymbol{x} = \frac{1}{N_q^2} \sum_{i=1}^{N_q} \sum_{j=1}^{N_q} \int G(\boldsymbol{x} - \boldsymbol{x}_{qi}, \sigma_q^2 \boldsymbol{I})\; G(\boldsymbol{x} - \boldsymbol{x}_{qj}, \sigma_q^2 \boldsymbol{I})\, d\boldsymbol{x} \tag{5.22}$$

Using the Gaussian convolution property and inserting the above equations into Eq. 5.19 yields:

$$\hat{D}_{CS}(p,q) = -\log \frac{\sum_{i=1}^{N_p} \sum_{j=1}^{N_q} G(\boldsymbol{x}_{pi} - \boldsymbol{x}_{qj}, \sigma_p^2 \boldsymbol{I} + \sigma_q^2 \boldsymbol{I})}{\sqrt{\sum_{i=1}^{N_p} \sum_{j=1}^{N_p} G(\boldsymbol{x}_{pi} - \boldsymbol{x}_{pj}, 2\sigma_p^2 \boldsymbol{I})\; \sum_{i=1}^{N_q} \sum_{j=1}^{N_q} G(\boldsymbol{x}_{qi} - \boldsymbol{x}_{qj}, 2\sigma_q^2 \boldsymbol{I})}} \tag{5.23}$$

Note that unlike in the case of other divergence measures presented above, in Eq. 5.23 the only approximation is the Parzen windowing itself, as due to the Gaussian convolution property there was no need to use Eq. 5.8. This suggests that potentially the estimator of D_{CS} should be more reliable than that of D_{KL}, D_J or D_{JS}. It is interesting to note, that D_{CS} can also be written as:

$$D_{CS}(p,q) = -\frac{1}{2}\left[H(X_p) + H(X_q) - 2\,H(X_p, X_q)\right] \tag{5.24}$$

where $H(X) = -\log IP(X)$ denotes Renyi's quadratic entropy and $IP(X)$ stands for the Information Potential [127], which emphasises the direct relation of D_{CS} to information theory.

The Cauchy–Schwarz divergence has been used e.g. for classification [84] and clustering [83].

5.3.5 Mean Integrated Squared Error

The Integrated Squared Error (ISE) is a measure of distance between two probability distributions. It is also a special case of a family of divergence measures presented in [89]. However, perhaps the best known application of ISE is estimation of kernel bandwidth in the Parzen density method [174]. ISE is given by:

$$ISE(p,q) = \int \left(p(\boldsymbol{x}) - q(\boldsymbol{x})\right)^2\, d\boldsymbol{x} \tag{5.25}$$

After rearranging and applying Eq. 5.8 the following estimation formula can be obtained:

$$\begin{aligned} ISE(p,q) &= \int p(\boldsymbol{x})\left[p(\boldsymbol{x}) - q(\boldsymbol{x})\right]\, d\boldsymbol{x} + \int q(\boldsymbol{x})\left[q(\boldsymbol{x}) - p(\boldsymbol{x})\right]\, d\boldsymbol{x} \\ &= E_{p(x)}\left[p(\boldsymbol{x}) - q(\boldsymbol{x})\right] + E_{q(x)}\left[q(\boldsymbol{x}) - p(\boldsymbol{x})\right] \\ &\approx \frac{1}{N_p}\sum_{i=1}^{N_p}\left[p(\boldsymbol{x}_{pi}) - q(\boldsymbol{x}_{pi})\right] + \frac{1}{N_q}\sum_{i=1}^{N_q}\left[q(\boldsymbol{x}_{qi}) - p(\boldsymbol{x}_{qi})\right] \end{aligned} \tag{5.26}$$

Using the Parzen window density estimators of Eqs. 5.2 and 5.3 and rearranging one gets:

$$\begin{aligned} \hat{ISE}(p,q) &= \frac{1}{N_p^2}\sum_{i=1}^{N_p}\sum_{j=1}^{N_p} G(\boldsymbol{x}_{pi}-\boldsymbol{x}_{pj},\sigma_p^2\boldsymbol{I}) + \frac{1}{N_q^2}\sum_{i=1}^{N_q}\sum_{j=1}^{N_q} G(\boldsymbol{x}_{qi}-\boldsymbol{x}_{qj},\sigma_q^2\boldsymbol{I}) \\ &- \frac{1}{N_pN_q}\sum_{i=1}^{N_p}\sum_{j=1}^{N_q} G(\boldsymbol{x}_{pi}-\boldsymbol{x}_{qj},\sigma_p^2\boldsymbol{I}) - \frac{1}{N_pN_q}\sum_{i=1}^{N_p}\sum_{j=1}^{N_q} G(\boldsymbol{x}_{pi}-\boldsymbol{x}_{qj},\sigma_q^2\boldsymbol{I}) \\ &= IP(X_p) + IP(X_q) - IP(X_p,X_q) - IP(X_q,X_p) \\ &\approx IP(X_p) + IP(X_q) - 2\,IP(X_p,X_q) \\ &\approx IP(X_p) + IP(X_q) - 2\,IP(X_q,X_p) \end{aligned} \tag{5.27}$$

which is a result surprisingly similar to Eq. 5.24, but this time the information potentials are used instead of entropies and the Gaussian kernel width is equal to σ rather than $\sqrt{2}\sigma$.

ISE can also be estimated using the kNN density estimators of Eqs. 5.5 and 5.6 yielding:

$$\begin{aligned} \tilde{ISE}(p,q) &= \frac{k\Gamma(d/2+1)}{\pi^{d/2}}\left[\frac{1}{N_p}\sum_{i=1}^{N_p}\frac{N_q s_k(\boldsymbol{x}_{pi})^d - (N_p-1)r_k(\boldsymbol{x}_{pi})^d}{(N_p-1)N_q r_k(\boldsymbol{x}_{pi})^d s_k(\boldsymbol{x}_{pi})^d}\right. \\ &+ \left.\frac{1}{N_q}\sum_{i=1}^{N_q}\frac{N_p r_k(\boldsymbol{x}_{qi})^d - (N_q-1)s_k(\boldsymbol{x}_{qi})^d}{(N_q-1)N_p r_k(\boldsymbol{x}_{qi})^d s_k(\boldsymbol{x}_{qi})^d}\right] \end{aligned} \tag{5.28}$$

Since ISE depends on the particular realisation of the N points [130, 174], in practice the Mean Integrated Squared Error (MISE) is used instead. MISE is simply the expectation of ISE and in the experiments described in Section 5.4 it is estimated using Eq. 5.8.

5.4 Empirical convergence of the divergence estimators

In this section, an empirical convergence study of the divergence measures described in Section 5.3, is presented.

5.4.1 Experiment setup

Following [124] an empirical convergence study using four toy problems has been designed. For each of them, two Gaussian distributions were used. The contour plots for the first three toy problems can be seen in Figure 5.1. The distributions were chosen to be Gaussian, that is $p(\boldsymbol{x}) = G(\boldsymbol{x}-\boldsymbol{\mu}_p,\ \boldsymbol{\Sigma}_p)$ and $q(\boldsymbol{x}) = G(\boldsymbol{x}-\boldsymbol{\mu}_q,\ \boldsymbol{\Sigma}_q)$, as in this case there exist closed–form formulas enabling exact calculations of most of the divergence measures introduced in the previous sections. The parameters of the distributions were:

1. Toy problem 1, which is the same as the problem used in [124]:

$$\begin{aligned} \boldsymbol{\mu}_p &= \begin{bmatrix}0\\0\end{bmatrix}, \qquad \boldsymbol{\Sigma}_p = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \\ \boldsymbol{\mu}_q &= \begin{bmatrix}0.5\\-0.5\end{bmatrix}, \quad \boldsymbol{\Sigma}_q = \begin{bmatrix}0.5 & 0.1\\0.1 & 0.3\end{bmatrix} \end{aligned} \tag{5.29}$$

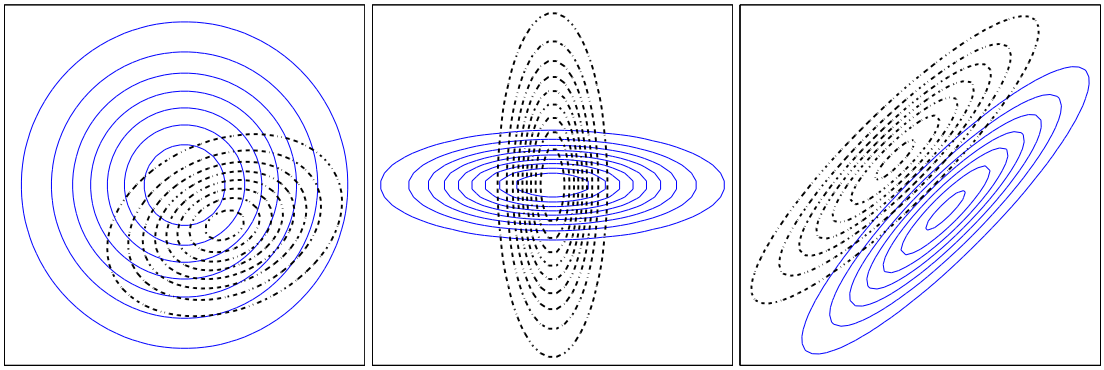
2. Toy problem 2, where the means are equal, but the covariance matrices are not:

$$\begin{aligned} \boldsymbol{\mu}_p &= \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{\Sigma}_p = \begin{bmatrix} 1 & 0 \\ 0 & 0.1 \end{bmatrix} \\ \boldsymbol{\mu}_q &= \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{\Sigma}_q = \begin{bmatrix} 0.1 & 0 \\ 0 & 1 \end{bmatrix} \end{aligned} \tag{5.30}$$

3. Toy problem 3, where the covariance matrices are equal, but the means are not:

$$\begin{aligned} \boldsymbol{\mu}_p &= \begin{bmatrix} 0.35 \\ -0.35 \end{bmatrix}, \quad \boldsymbol{\Sigma}_p = \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix} \\ \boldsymbol{\mu}_q &= \begin{bmatrix} -0.35 \\ 0.35 \end{bmatrix}, \quad \boldsymbol{\Sigma}_q = \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix} \end{aligned} \tag{5.31}$$

4. Toy problem 4, where the Gaussians are 20–dimensional, $\boldsymbol{\mu}_p = \begin{bmatrix} 0 & 0 & \dots & 0 \end{bmatrix}^T$, $\boldsymbol{\Sigma}_p = \boldsymbol{I}$ and $\boldsymbol{\mu}_q$, $\boldsymbol{\Sigma}_q$ have been generated randomly from the $[-1, +1]$ and $[0, +2]$ intervals respectively.



(a) Toy problem 1 (b) Toy problem 2 (c) Toy problem 3

Figure 5.1: Contour plots for the toy problems: solid line – $p(\boldsymbol{x})$, dotted line – $q(\boldsymbol{x})$

For each experiment 100 random samples of an exponentially increasing size were drawn from both $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$. The divergence estimate was then calculated as the mean value of estimates for each of these 100 samples.

Denoting by d the dimensionality of the distributions, the Kullback–Leibler divergence between two Gaussian distributions $p_G(\boldsymbol{x})$ and $q_G(\boldsymbol{x})$ can be calculated using the following formula [184]:

$$D_{KL}(p_G, q_G) = \frac{1}{2}\left(\log\left(\frac{\det \boldsymbol{\Sigma}_q}{\det \boldsymbol{\Sigma}_p}\right) + Tr\left(\boldsymbol{\Sigma}_q^{-1}\boldsymbol{\Sigma}_p\right) + \left(\boldsymbol{\mu}_q - \boldsymbol{\mu}_p\right)^T \boldsymbol{\Sigma}_q^{-1}\left(\boldsymbol{\mu}_q - \boldsymbol{\mu}_p\right) - d\right) \tag{5.32}$$

Calculation of the Jeffrey's divergence between two Gaussian PDFs is straightforward, as $D_J(p_G, q_G) = D_{KL}(p_G, q_G) + D_{KL}(q_G, p_G)$.

In order to calculate D_{CS} and ISE exactly for the special case of two Gaussian distributions, the following Gaussian multiplication formula is used:

$$G(\boldsymbol{x}-\boldsymbol{\mu}_p,\ \boldsymbol{\Sigma}_p)G(\boldsymbol{x}-\boldsymbol{\mu}_q,\ \boldsymbol{\Sigma}_q) = G(\boldsymbol{\mu}_p-\boldsymbol{\mu}_q,\ \boldsymbol{\Sigma}_p+\boldsymbol{\Sigma}_q)G(\boldsymbol{x}-\boldsymbol{\mu}_{pq},\ \boldsymbol{\Sigma}_{pq}) \tag{5.33}$$

where the exact expression for $\boldsymbol{\mu}_{pq}$ and $\boldsymbol{\Sigma}_{pq}$ in this case are irrelevant, as shown below. Using Eq. 5.33, the following holds:

$$\int p_G^2(\boldsymbol{x})\,d\boldsymbol{x} = \int G(\boldsymbol{\mu}_p-\boldsymbol{\mu}_p,\ \boldsymbol{\Sigma}_p+\boldsymbol{\Sigma}_p)\,G(\boldsymbol{x}-\boldsymbol{\mu}_{pp},\ \boldsymbol{\Sigma}_{pp})\,d\boldsymbol{x} = G(0,\ 2\,\boldsymbol{\Sigma}_p) \tag{5.34}$$

$$\int q_G^2(\boldsymbol{x})\,d\boldsymbol{x} = \int G(\boldsymbol{\mu}_q-\boldsymbol{\mu}_q,\ \boldsymbol{\Sigma}_q+\boldsymbol{\Sigma}_q)\,G(\boldsymbol{x}-\boldsymbol{\mu}_{qq},\ \boldsymbol{\Sigma}_{qq})\,d\boldsymbol{x} = G(0,\ 2\,\boldsymbol{\Sigma}_q) \tag{5.35}$$

$$\int p_G(\boldsymbol{x})q_G(\boldsymbol{x})\,d\boldsymbol{x} = \int G(\boldsymbol{\mu}_p-\boldsymbol{\mu}_q,\ \boldsymbol{\Sigma}_p+\boldsymbol{\Sigma}_q)\,G(\boldsymbol{x}-\boldsymbol{\mu}_{pq},\ \boldsymbol{\Sigma}_{pq})\,d\boldsymbol{x} = G(\boldsymbol{\mu}_p-\boldsymbol{\mu}_q,\ \boldsymbol{\Sigma}_p+\boldsymbol{\Sigma}_q) \tag{5.36}$$

From the above and Eqs. 5.19 and 5.25 the closed–form solutions for D_{CS} and ISE are:

$$D_{CS}(p_G,q_G) = -\log\frac{G(\boldsymbol{\mu}_p-\boldsymbol{\mu}_q,\ \boldsymbol{\Sigma}_p+\boldsymbol{\Sigma}_q)}{\sqrt{G(0,\ 2\,\boldsymbol{\Sigma}_p)G(0,\ 2\,\boldsymbol{\Sigma}_q)}} \tag{5.37}$$

$$\begin{aligned} ISE(p_G,q_G) &= \int p_G^2(\boldsymbol{x})\,d\boldsymbol{x} + \int q_G^2(\boldsymbol{x})\,d\boldsymbol{x} - 2\int p_G(\boldsymbol{x})q_G(\boldsymbol{x})\,d\boldsymbol{x} \\ &= G(0,\ 2\,\boldsymbol{\Sigma}_p) + G(0,\ 2\,\boldsymbol{\Sigma}_q) - 2\,G(\boldsymbol{\mu}_p-\boldsymbol{\mu}_q,\ \boldsymbol{\Sigma}_p+\boldsymbol{\Sigma}_q) \end{aligned} \tag{5.38}$$

Unfortunately, there is no closed–form formula to calculate the Jensen–Shannon divergence, and an estimate based on Eq. 5.7 has been used, calculated for $N_p = N_q = 100000$, yielding:

$$D_{JS}(p_G,q_G) = \frac{1}{2}D_{KL}(p_G,\frac{1}{2}(p_G+q_G)) + \frac{1}{2}D_{KL}(q_G,\frac{1}{2}(p_G+q_G)) \tag{5.39}$$

where:

$$D_{KL}(p_G,\frac{1}{2}(p_G+q_G)) \approx \frac{1}{N_p}\sum_{i=1}^{N_p}\log\frac{p_G(\boldsymbol{x}_{pi})}{\frac{1}{2}(p_G(\boldsymbol{x}_{pi})+q_G(\boldsymbol{x}_{pi}))} \tag{5.40}$$

$$D_{KL}(q_G,\frac{1}{2}(p_G+q_G)) \approx \frac{1}{N_q}\sum_{i=1}^{N_q}\log\frac{q_G(\boldsymbol{x}_{pi})}{\frac{1}{2}(p_G(\boldsymbol{x}_{qi})+q_G(\boldsymbol{x}_{qi}))} \tag{5.41}$$

Note, that in both equations above the terms under the logarithms are not dependent on any PDF estimator–specific parameters, as the PDFs are given in analytical forms, so the sample mean is the only estimation required.

Figures 5.2 to 5.10 present the results of the experiments using the 4 toy problems. The 'best' estimate in each plot has been marked with bold red line. Additionally the confidence intervals (mean $+/-$ one standard deviation) for the best method were also plotted. The best estimator has been chosen according to the criterion of fast convergence to the true value of the estimated measure and lack of divergence after reaching that target value. To facilitate the choice of

the 'best' estimator the below scoring function was used:

$$S = \left(\sum_{i=1}^{|\mathcal{M}|} m_i \left(\bar{y}_i - t \right)^2 \right)^{-1} \tag{5.42}$$

where $\mathcal{M} = \{10, 20, \ldots, 100, 200, \ldots, 1000, 2000, \ldots, 10000\}$, $\bar{y}_i$ is the mean value of the estimator for the sample size m_i and t is the true value of the divergence measure. Note that this scoring function heavily penalizes any deviation from the true value for large sample sizes, which in effect assigns low scores to estimators which have not converged or started to diverge.

In the Figures below the following code has been used for denoting the divergence estimators:

$XX - YYZZ$	for Parzen window density estimates
$XX - k$	for kNN density estimates

where k stands for the number of nearest neighbours and the remaining symbols have been given in Tables 5.1 to 5.3.

Table 5.1: Divergence measure estimators

XX	**description**
kl1	Kullback–Leibler divergence estimator based on Parzen window density
kl2	Kullback–Leibler divergence estimator based on kNN density
j1	Jeffrey's divergence estimator based on Parzen window density
j2	Jeffrey's divergence estimator based on kNN density
js1	Jensen–Shannon divergence estimator based on Parzen window density
js2	Jensen–Shannon divergence estimator based on kNN density
cs	Cauchy–Schwarz divergence estimator
ise1	Integrated Squared Error estimator based on Parzen window density
ise2	Integrated Squared Error estimator based on kNN density

Table 5.2: Automatic bandwidth selection methods

YY	**description**
ml	Pseudo–likelihood cross–validation
rot	Rule of Thumb
amise	AMISE minimisation

Table 5.3: Gaussian kernel covariance matrix

ZZ	description
1c	Identity covariance matrix multiplied by a scalar, common for both distributions
2c	Diagonal covariance matrix, common for both distributions
1s	Identity covariance matrix multiplied by a scalar, separate for each distribution
2s	Diagonal covariance matrix, separate for each distribution

5.4.2 Estimation of the Kullback–Leibler divergence

The plots presenting the evolution of Kullback–Leibler divergence estimators based on the Parzen window method as the sample size increases, for all 4 toy problems, have been given in Figure 5.2. The values on the horizontal axis denote the number of instances drawn from each distribution. As it can be seen, there is a considerable discrepancy between various estimators (i.e. estimators using various bandwidth selection methods). More specifically, while some of them seem to be converging[1] to the true value, others diverge, which is especially well visible in the case of high–dimensional toy problem 4. Moreover, even the 'best' estimators reach the true divergence values for sample sizes, for which, if encountered in practice, the representativeness of even a random subsample should not be a problem. In such cases the purposefulness of divergence guided sampling seems doubtful.

Figure 5.3 presents the experimental results for Kullback–Leibler divergence estimators based on the kNN density estimator. In this case, the convergence for the 2–dimensional problems, albeit slow, can still be observed regardless of the value of k and has also been proven in [124]. However, for the 20–dimensional problem 4, the estimators fail completely.

5.4.3 Estimation of the Jeffrey's divergence

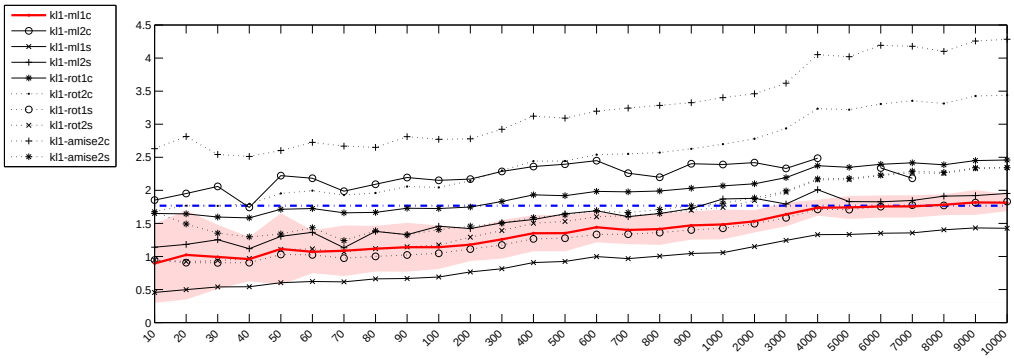
The behaviour of the Parzen window based estimators of the Jeffrey's divergence have been depicted in Figure 5.4. For the 2–dimensional problems, the picture is very similar to the case of D_{KL}. However, in the high–dimensional space most of the estimators cannot be evaluated due to numerical problems, resulting from near–zero values of many Gaussian functions in calculation of $\hat{D}_{KL}(q, p)$.

The Jeffrey's divergence estimator based on kNN density depicted in Figure 5.5 also behaves similarly to $\tilde{D}_{KL}$. Although no numerical problems have been observed in the high–dimensional scenario, the estimators are off the true divergence value by a large margin.

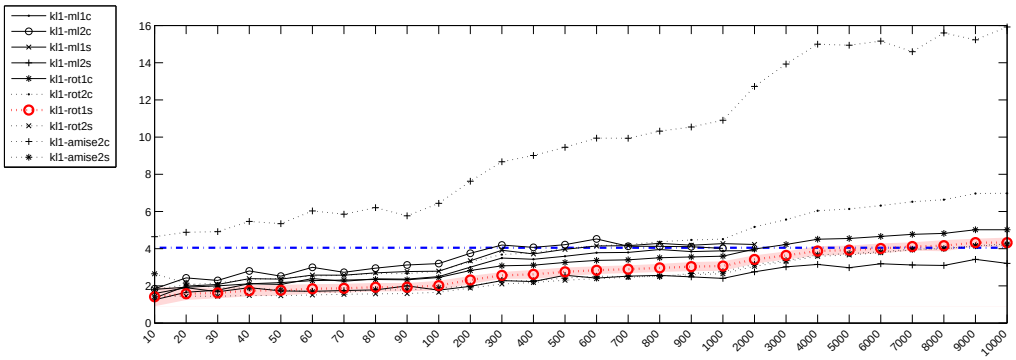
5.4.4 Estimation of the Jensen–Shannon's divergence

The experimental results for the Jensen–Shannon's divergence estimators given in Figures 5.6 and 5.7, look much more promising. The convergence of the Parzen window based estimators is rapid when compared to $\hat{D}_{KL}$ and $\hat{D}_J$, as it takes place for sample sizes of 400–500. What is even more important, the estimators, regardless of the bandwidth selection method used are usually in agreement when the shapes of the convergence curves are taken into account (the exception is

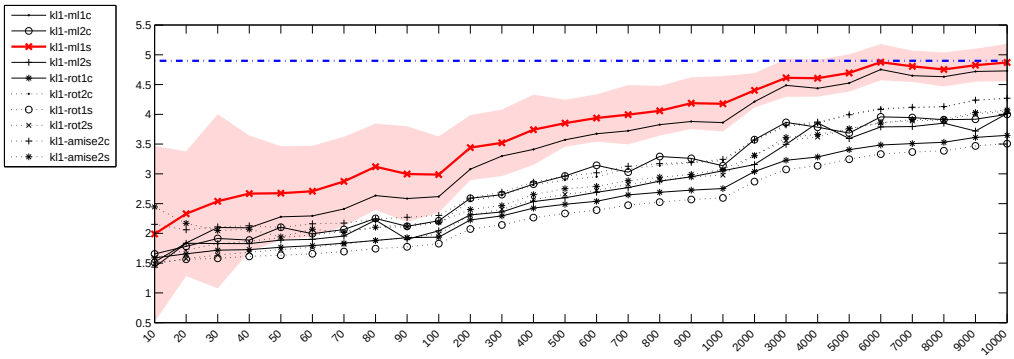
[1] since the apparent convergence takes place for sample sizes of 5000–6000 (depending on the problem), one cannot be sure what would happen for samples larger than 10000, which have not been examined here due to high computational requirements



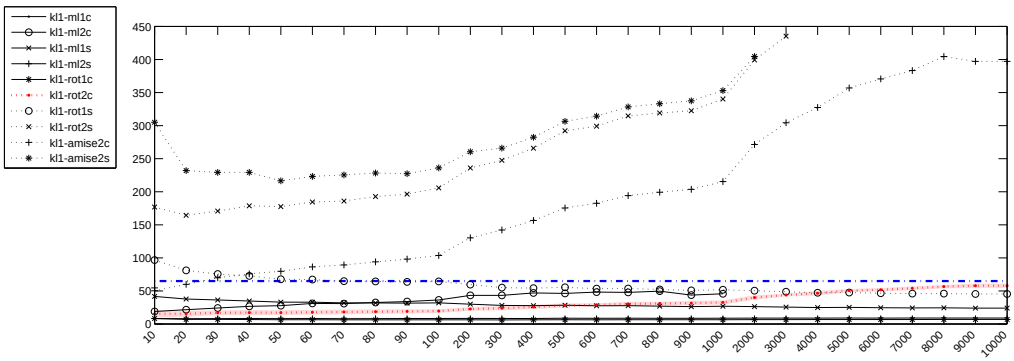
(a) Toy problem 1



(b) Toy problem 2

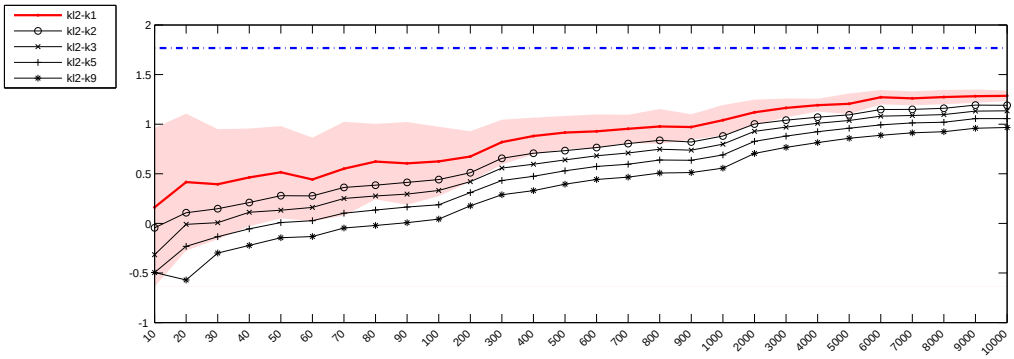


(c) Toy problem 3

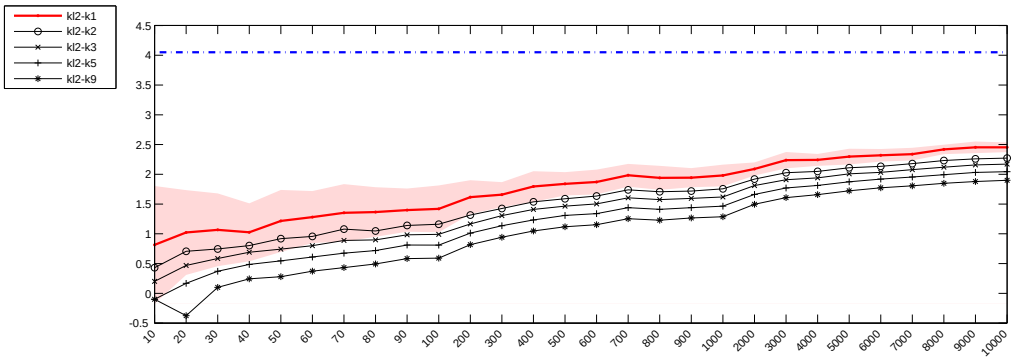


(d) Toy problem 4

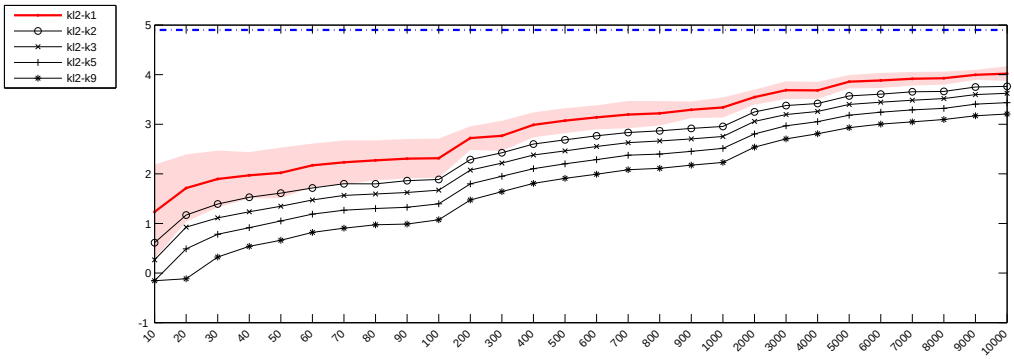
Figure 5.2: Parzen density based Kullback–Leibler divergence estimator $\hat{D}_{KL}$ (horizontal axis – sample size, vertical axis – estimated value)



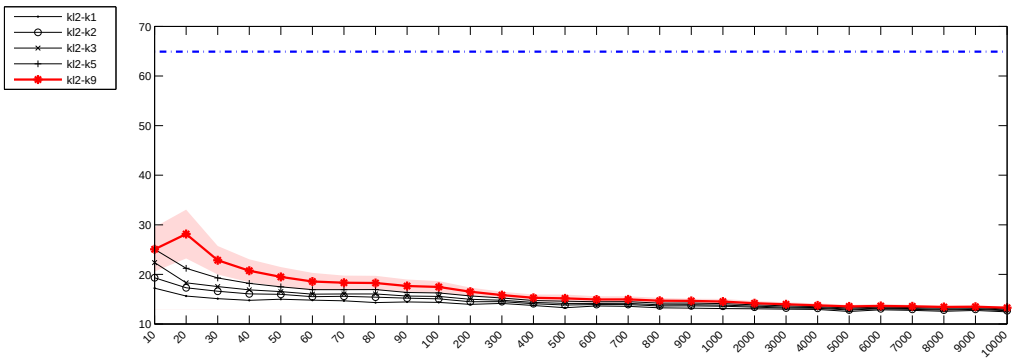
(a) Toy problem 1



(b) Toy problem 2

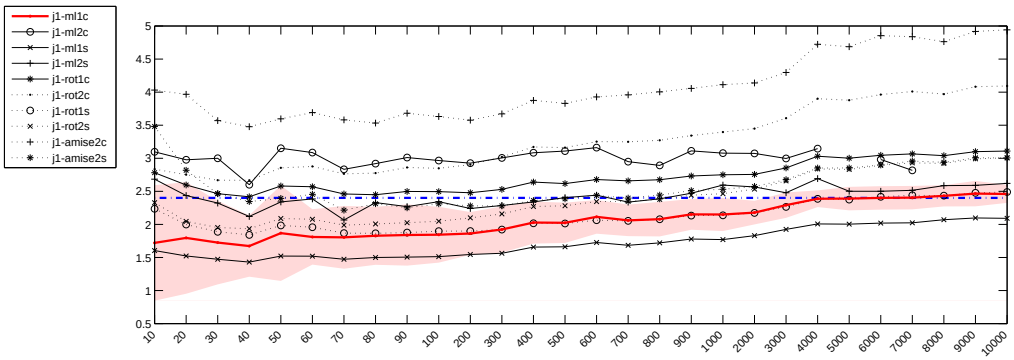


(c) Toy problem 3

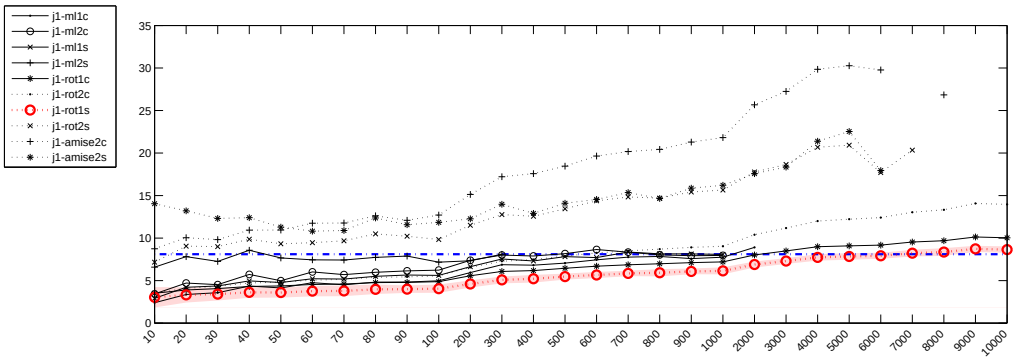


(d) Toy problem 4

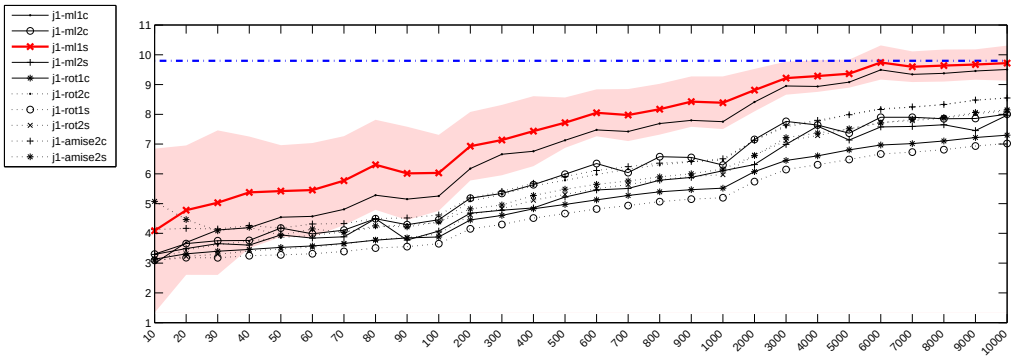
Figure 5.3: kNN density based Kullback–Leibler divergence estimator $\tilde{D}_{KL}$ (horizontal axis – sample size, vertical axis – estimated value)



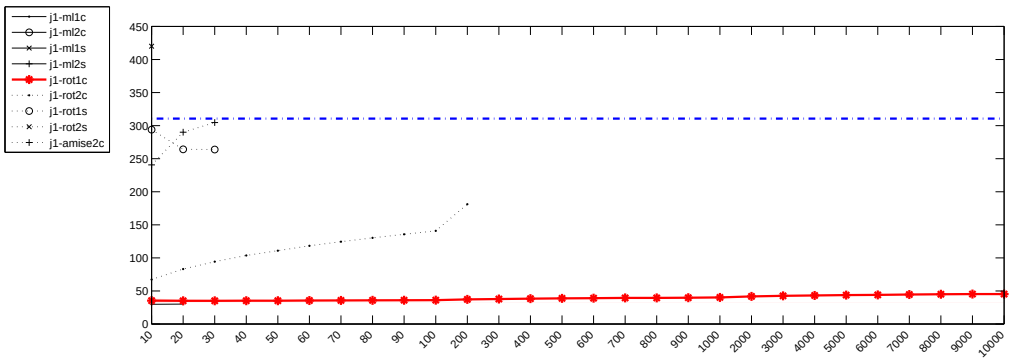
(a) Toy problem 1



(b) Toy problem 2

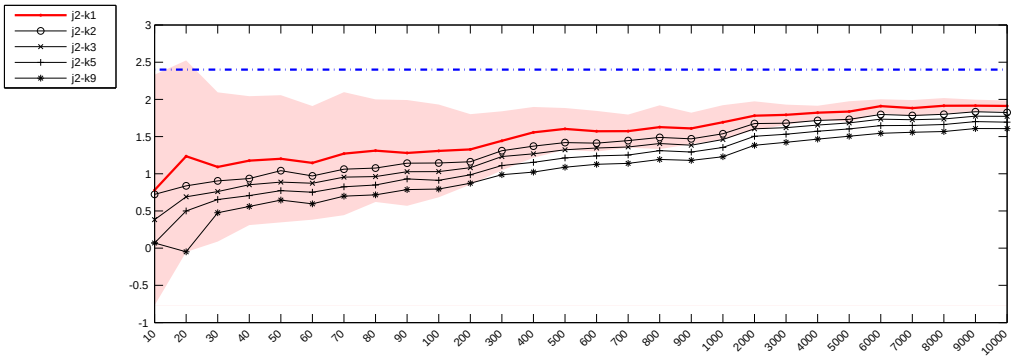


(c) Toy problem 3

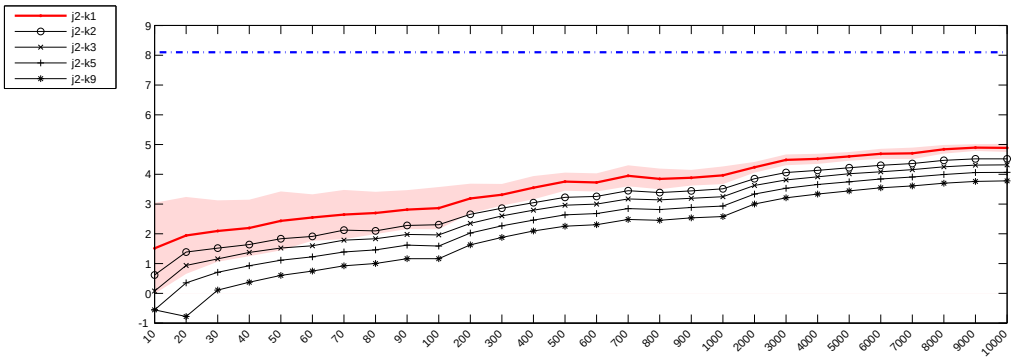


(d) Toy problem 4

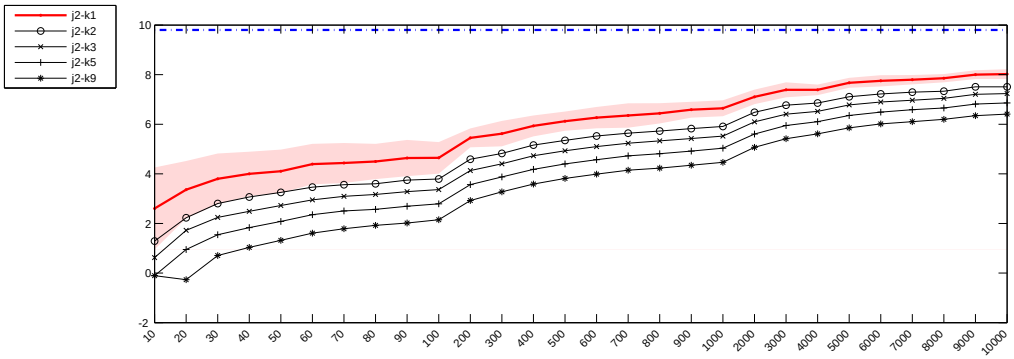
Figure 5.4: Parzen density based Jeffrey's divergence estimator $\hat{D}_J$ (horizontal axis – sample size, vertical axis – estimated value)



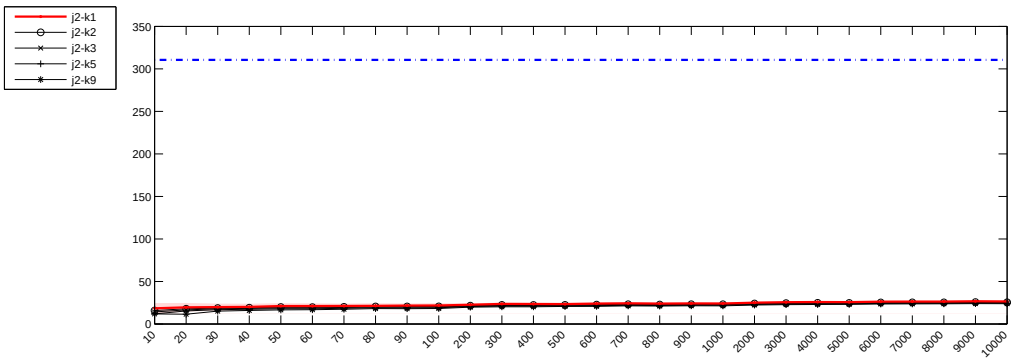
(a) Toy problem 1



(b) Toy problem 2

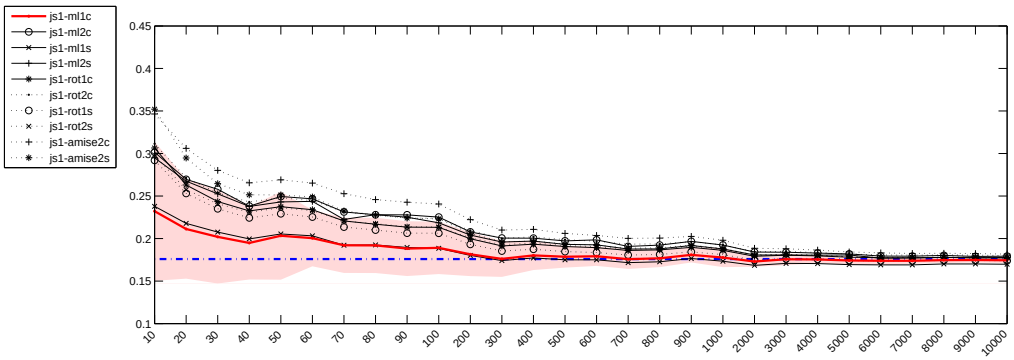


(c) Toy problem 3

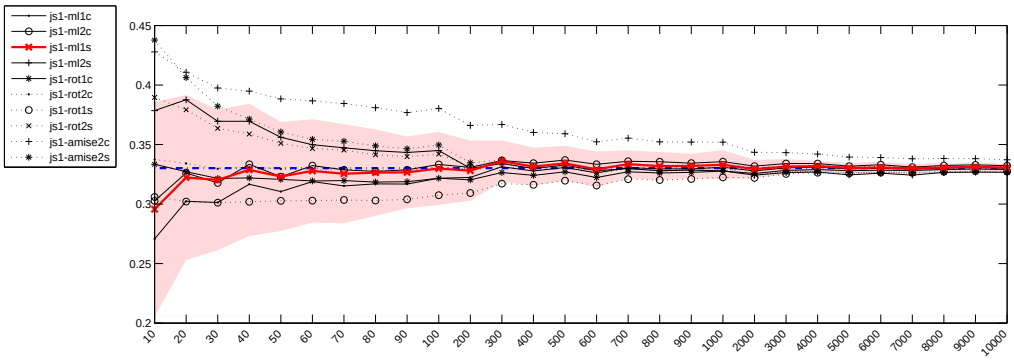


(d) Toy problem 4

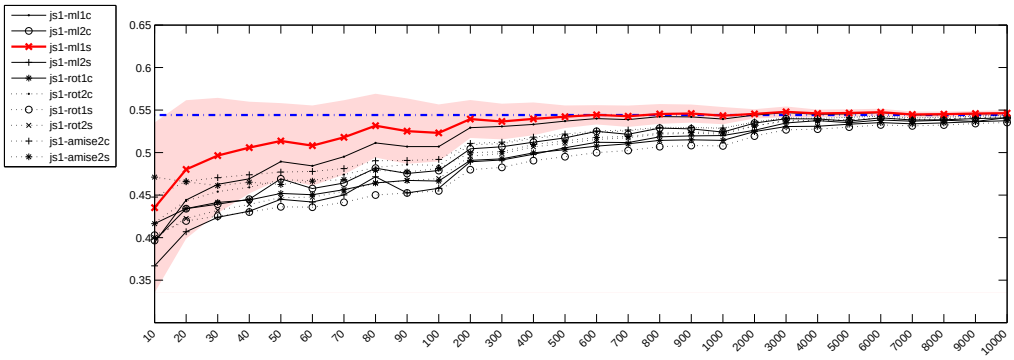
Figure 5.5: kNN density based Jeffrey's divergence estimator $\tilde{D}_J$ (horizontal axis – sample size, vertical axis – estimated value)



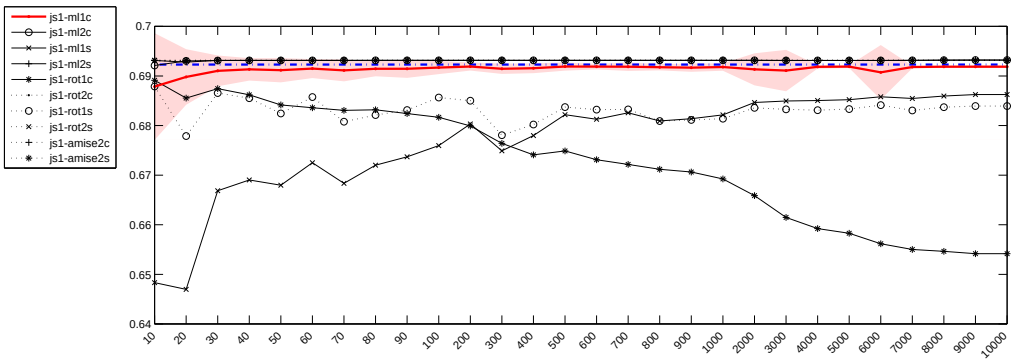
(a) Toy problem 1



(b) Toy problem 2

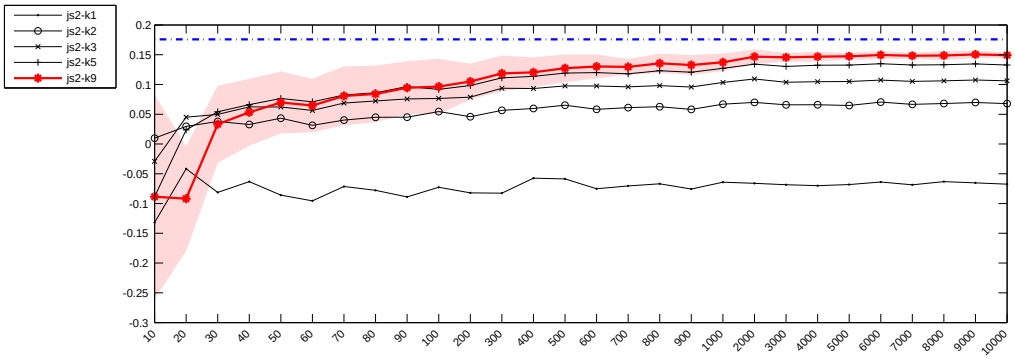


(c) Toy problem 3

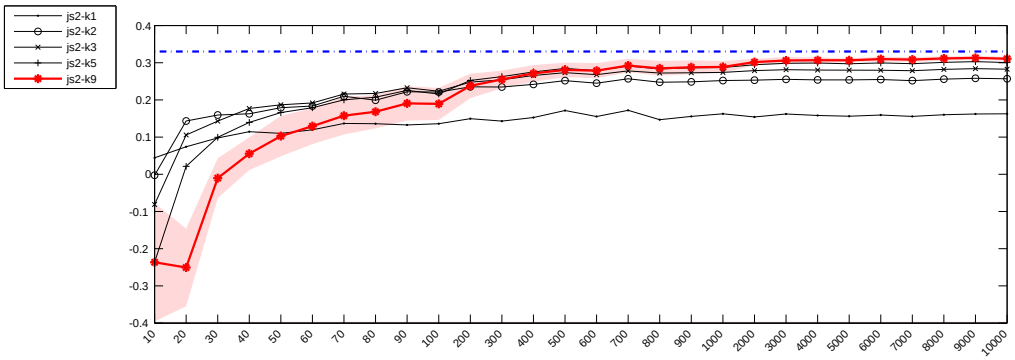


(d) Toy problem 4

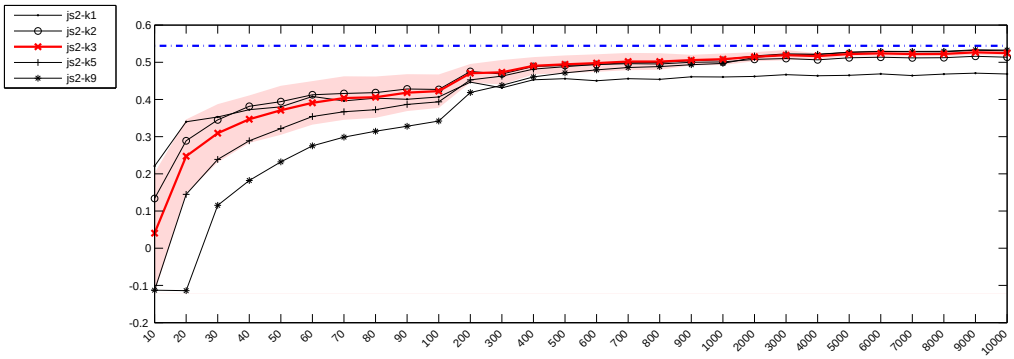
Figure 5.6: Parzen density based Jensen–Shannon's divergence estimator $\hat{D}_{JS}$ (horizontal axis – sample size, vertical axis – estimated value)



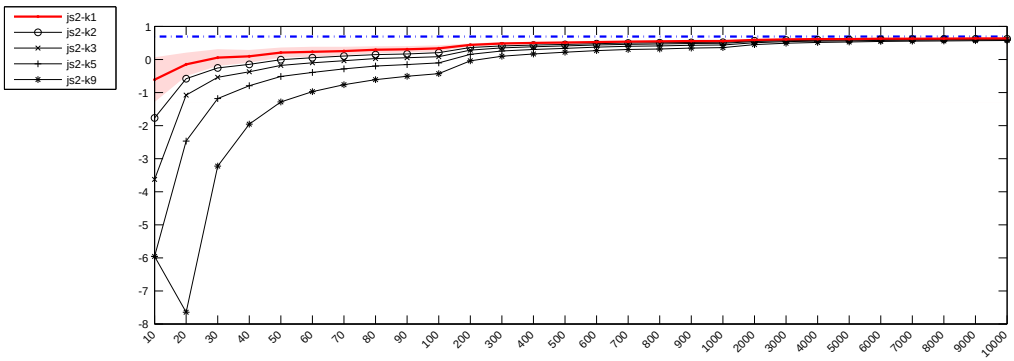
(a) Toy problem 1



(b) Toy problem 2



(c) Toy problem 3



(d) Toy problem 4

Figure 5.7: kNN density based Jensen–Shannon's divergence estimator $\tilde{D}_{JS}$ (horizontal axis – sample size, vertical axis – estimated value)

the 20–dimensional problem). It should however be kept in mind, that no closed–form formula for calculation of the true value exists, so effectively in this setting an estimate of the true value is approximated.

Although most of the kNN density based estimators also seem to be converging to the true value, the convergence is considerably slower than in the case of their Parzen window based counterparts. This can be clearly seen by examining the units on the vertical axis in Figures 5.6 and 5.7.

5.4.5 Estimation of the Integrated Squared Error

The convergence curves for the Integrated Squared Error estimates have been depicted in Figures 5.8 and 5.9. For the first two toy problems, the Parzen window based estimators behave in a desired way, relatively quickly approaching the true value of ISE. The situation looks a bit different in case of the toy problem 3, where for the examined sample sizes none of the estimators approaches the true value within 0.05. An interesting situation has however developed in the case of toy problem 4 – throughout the whole range of sample sizes most estimators are very close to the true value, which at 1.0184e–011 is itself very close to 0 and can pose numerical problems.

The situation for kNN density based estimators looks even more interesting. For small values of k (1 and 2) the estimator is unstable and its values vary greatly with the sample size. For this reason the $k = 1$ case has not been included in the plots. The estimators also in general diverge and behave suspiciously in the 20–dimensional case.

5.4.6 Estimation of the Cauchy–Schwarz divergence

The experimental results for the estimation of the Cauchy–Schwarz divergence can be seen in Figure 5.10. Although as mentioned in Section 5.3.4, as opposed to other divergence measures, in this case the only approximation is the Parzen windowing itself (no need to use Eq. 5.8), the behaviour of $\hat{D}_{CS}$ is not as good as one would expect. More specifically, the estimator did not reach the true value even for a sample of 10000 instances in the case of toy problem 3 and has diverged in the 20–dimensional scenario.

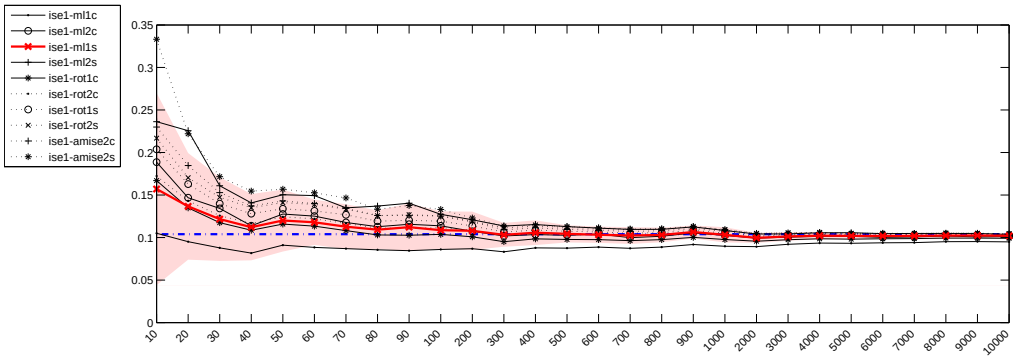
5.4.7 Summary

The picture emerging from the experimental results does not look very optimistic. Many estimates of various divergence measures either diverge or converge too slowly, questioning their usefulness for the purpose pursued in this study. From all the estimators examined, the Parzen window based Jensen–Shannon's divergence estimator looks most promising, as it converges relatively quickly although it also demonstrates a considerable[2] variance before converging.

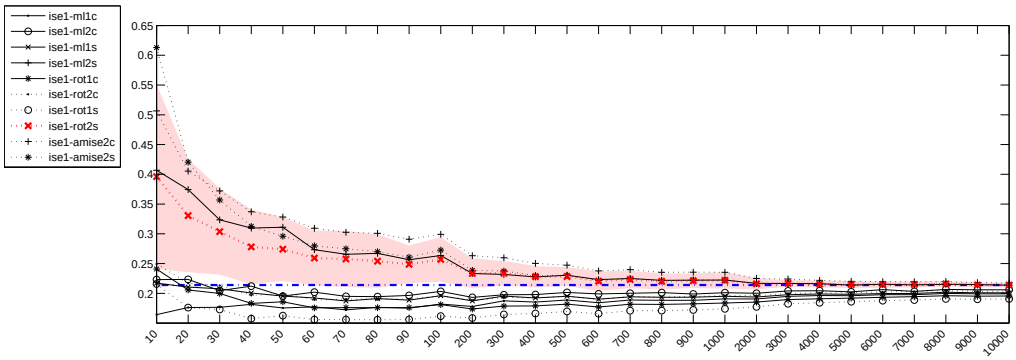
A common problem is also the behaviour of most estimators in a high–dimensional space. In this case only $\hat{D}_{JS}$ and $\tilde{D}_{JS}$ seem to be acting reasonably (once again, note the units on the vertical axis in Figures 5.6(d) and 5.7(d)) but in general Parzen windowing with kernels with exponential fall–off (e.g. Gaussian) is known to be problematic, as the density in large areas of the high–dimensional space is necessarily close to 0.

It is also important to have in mind that all the toy problems examined are relatively simple, as they consist of symmetric, unimodal distributions, which unfortunately are rarely encountered in practice.

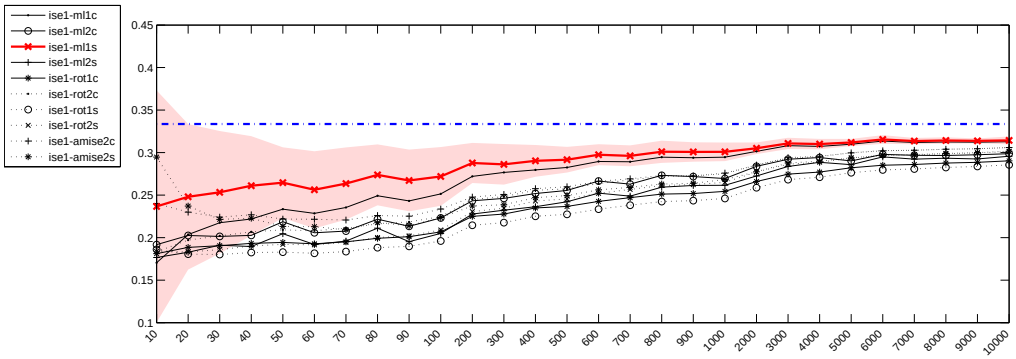
[2] even for a sample of 10 instances the true value is within one standard deviation from the mean value of the estimate



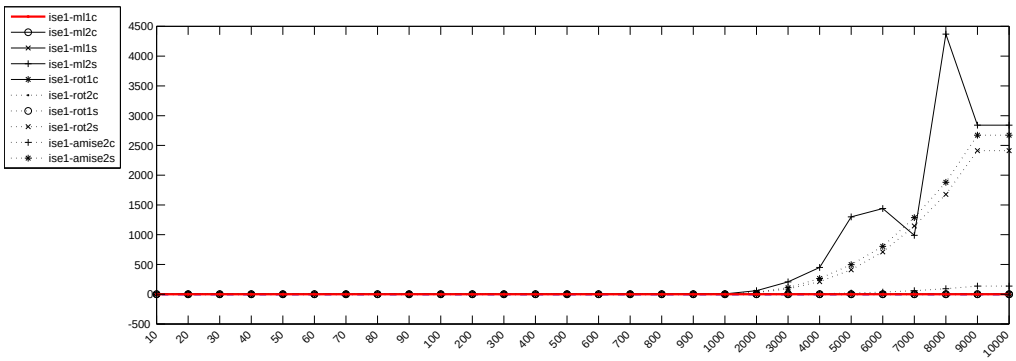
(a) Toy problem 1



(b) Toy problem 2

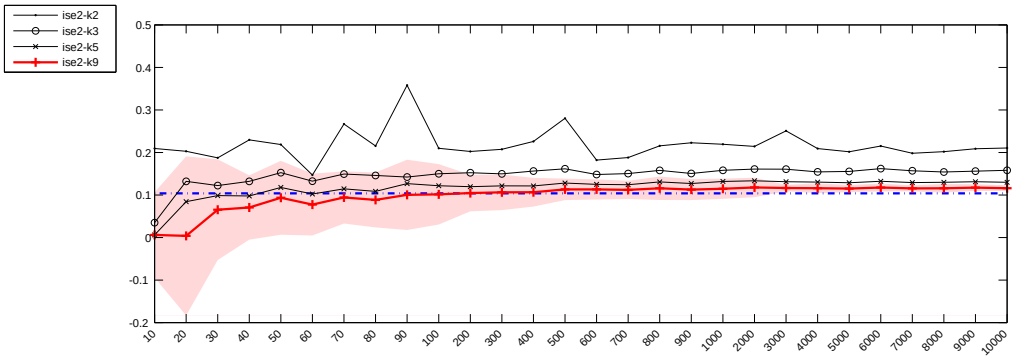


(c) Toy problem 3

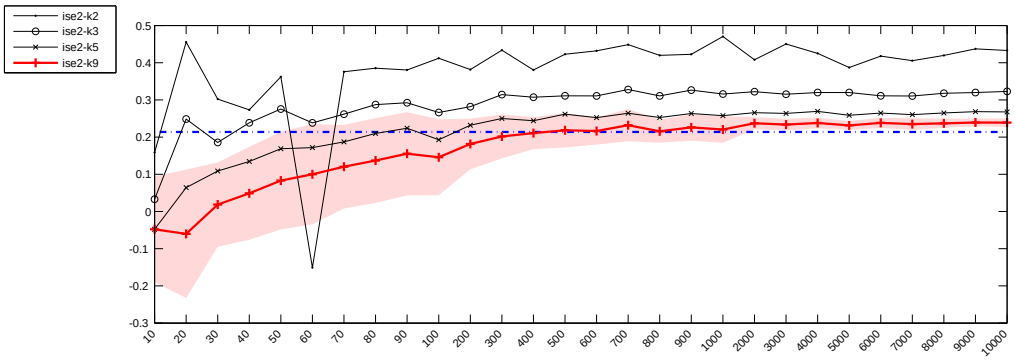


(d) Toy problem 4

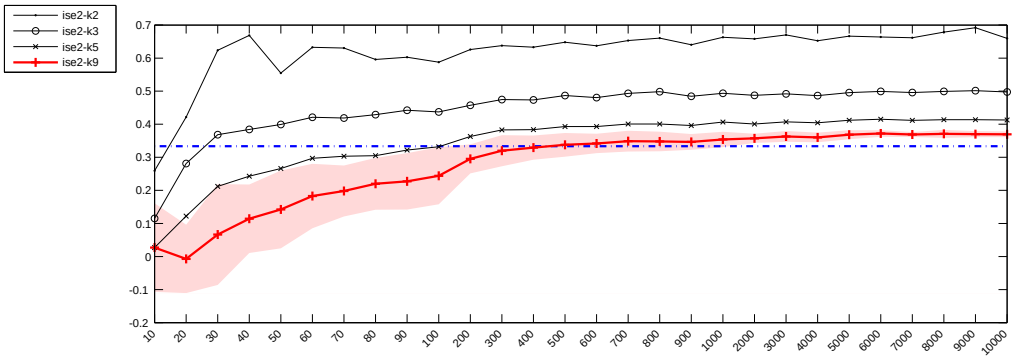
Figure 5.8: Parzen density based ISE estimator $\hat{ISE}$ (horizontal axis – sample size, vertical axis – estimated value)



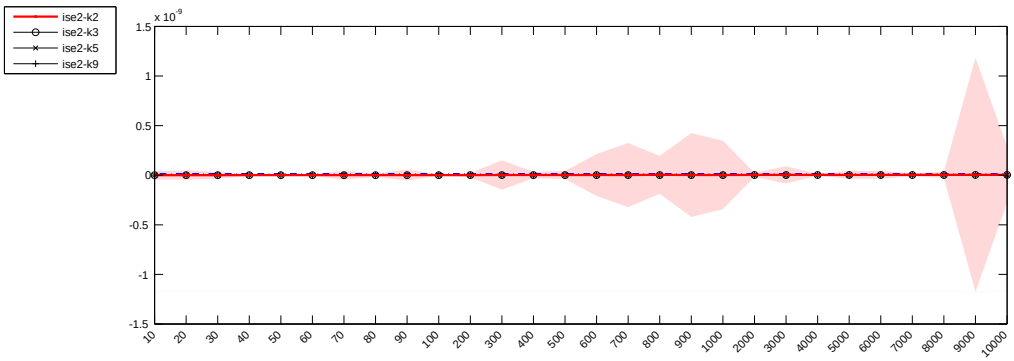
(a) Toy problem 1



(b) Toy problem 2

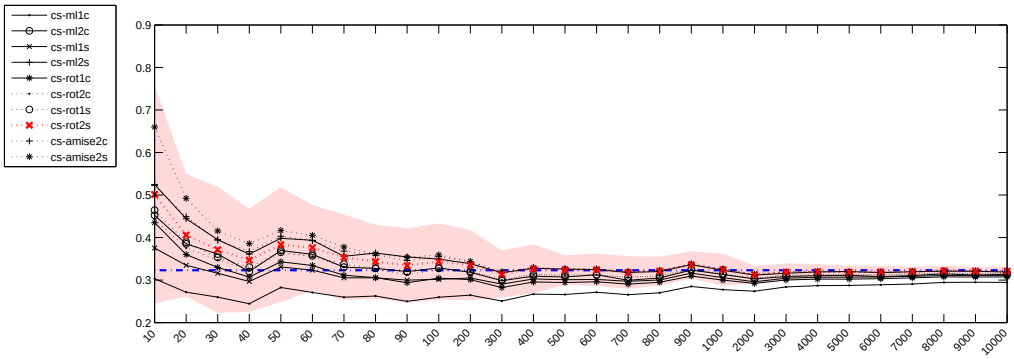


(c) Toy problem 3

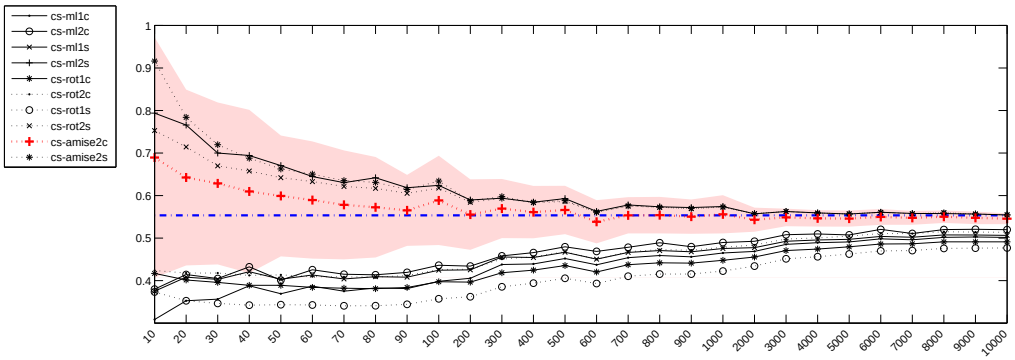


(d) Toy problem 4

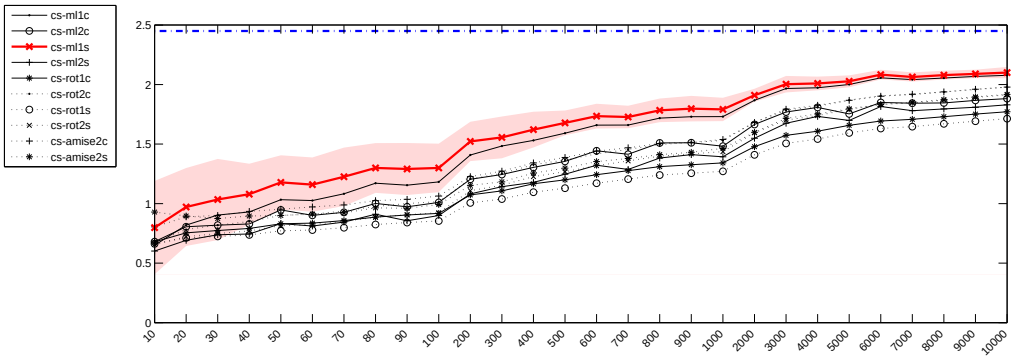
Figure 5.9: kNN density based ISE estimator $I\tilde{S}E$ (horizontal axis – sample size, vertical axis – estimated value)



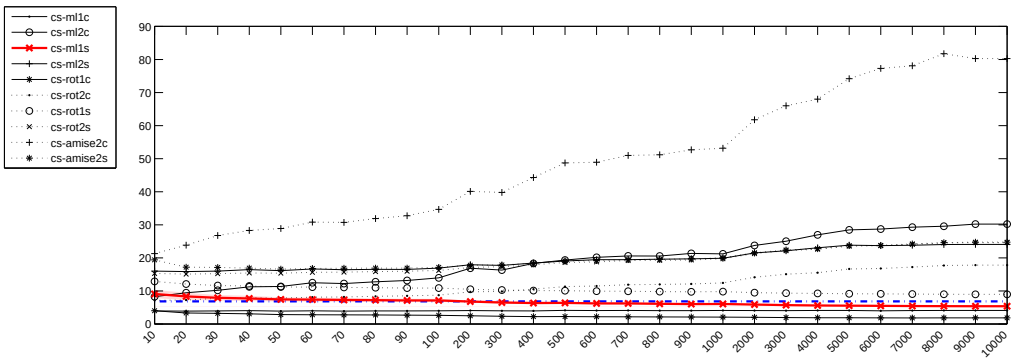
(a) Toy problem 1



(b) Toy problem 2



(c) Toy problem 3



(d) Toy problem 4

Figure 5.10: Parzen density based Cauchy–Schwarz divergence estimator $\hat{D}_{CS}$ (horizontal axis – sample size, vertical axis – estimated value)

5.5 PDF divergence guided sampling for error estimation

In this section an empirical study of correlation between various PDF divergence estimators and bias of a generalisation error estimate is investigated using a number of benchmark datasets. As already discussed, the goal of this experiments is to assess the possibility of estimating the generalisation error in a single run, i.e. without retraining the used model. This would allow to further reduce the computational cost of error estimation when compared to both CV and DPS.

5.5.1 Experiment setup

The experiments have been performed using 26 publicly available datasets and 18 different classifiers, including GFC and EFC discussed in Section 3.2. The datasets used have been described in Appendix A (one of them has been excluded from experiments due to numerical problems it was causing during automatic Parzen window bandwidth selection). The list of classifiers can be found in Appendix B.1. For each dataset the following procedure was followed:

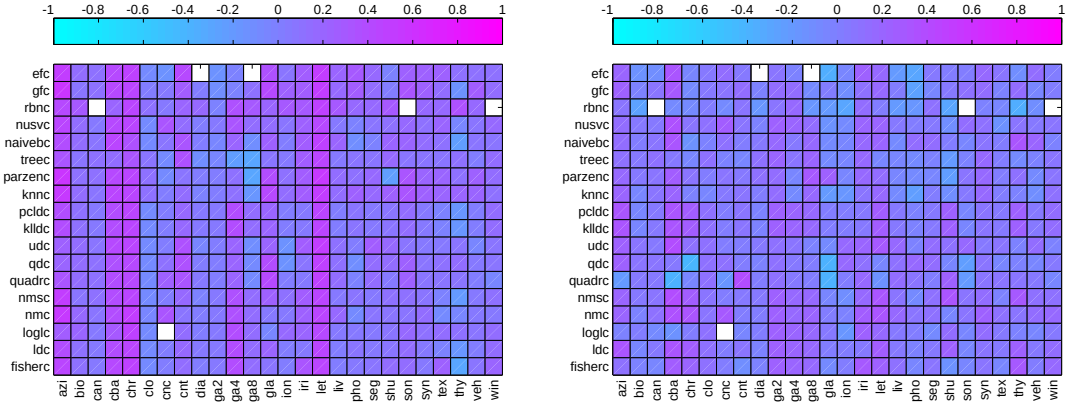
1. 400 stratified splits of the dataset have been generated, leaving 87.5% of instances for training and 12.5% instances for testing,
2. for each split and each class of the dataset, 70 different estimators of divergence between the training and test parts have been calculated, accounting for all possible combinations of techniques listed in Tables 5.1 to 5.3, where in the case of kNN density based divergence estimators $k = \{1, 2, 3, 5, 9\}$ were used,
3. for each divergence estimator the classes have been sorted by the estimated value, forming 400 new splits per estimator (since for some splits some estimators produced negative values, these splits have been discarded),
4. 11 splits have been selected for each divergence estimator based on the estimated value averaged over all classes, including the splits for the lowest and highest averaged estimate, as well as 9 splits for intermediate values,
5. the classifiers have been trained using the training part and tested using the test part for each split and each divergence estimator, producing error estimates sorted according to the divergence estimate.

5.5.2 Correlation between divergence estimators and bias

In the course of the experiments over 30000 correlation coefficients have been calculated, accounting for all dataset/classifiers/divergence estimator triplets, with the exception of the cases in which calculation of correlation was not possible due to numerical problems of the divergence estimators (especially the ones based on AMISE Parzen window bandwidth selection).

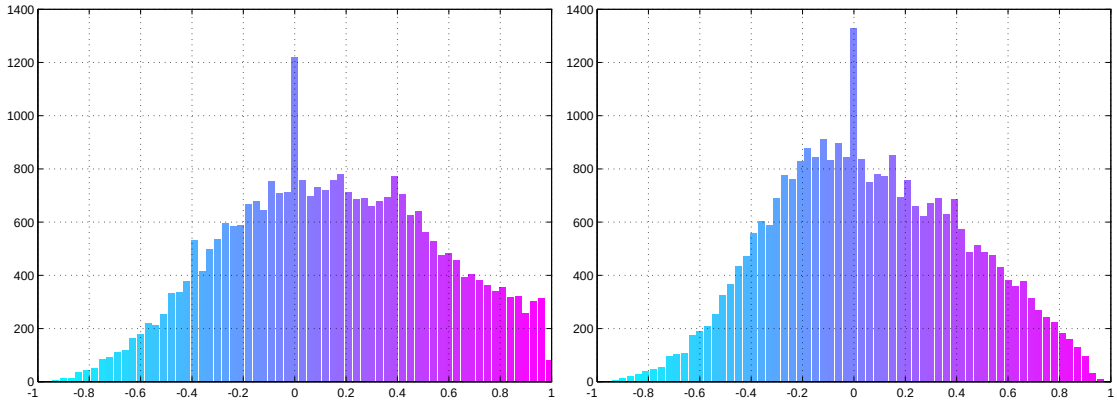
The maps of linear correlation coefficients between bias and divergence estimates, averaged over all divergence estimators used in the experiments have been depicted in Figure 5.11. The blank spots denote the situation in which for all 11 splits both the values of divergence estimator and bias were constant, so it was impossible to assess correlation. As it can be seen, for the signed bias moderate correlation can be observed only for a handful of datasets. However, in some cases this applies to all (*chr*, *let*) or almost all (*cba*) classifiers. For other datasets the correlation is weak to none and sometimes even negative. Only occasional and rather weak correlation can be observed in the absolute bias scenario. This can seen in Figure 5.12, which presents histograms of correlation coefficients for all 30k dataset/classifier/divergence estimator

triplets in both signed and absolute bias scenarios. As it can be seen, only the former scenario appears viable, as in the case of absolute bias the histogram is skewed towards -1 rather than $+1$.



(a) Signed bias (b) Absolute bias

Figure 5.11: Correlation between bias and divergence estimates averaged over the latter

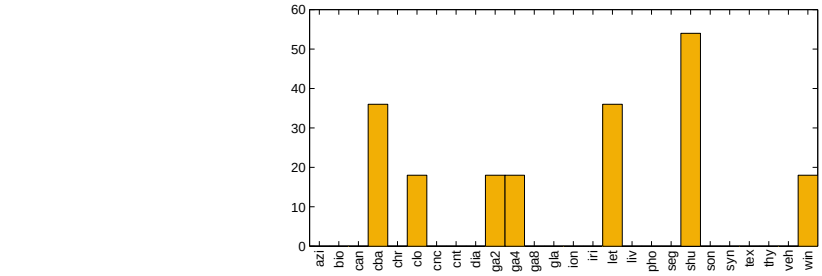


(a) Signed bias (b) Absolute bias

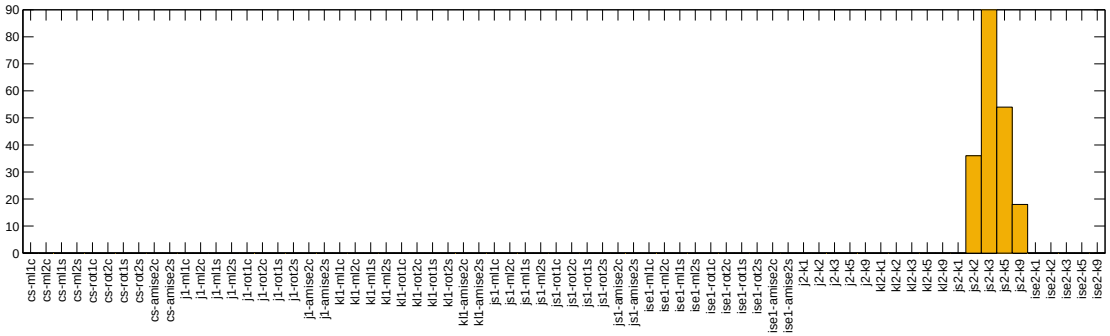
Figure 5.12: Correlation coefficient histograms for all dataset/classifier/divergence estimator triplets

One thing that requires explanation with respect to Figure 5.12 is the height of the bars centered at 0, for both signed and absolute bias. The bars represent cases where the correlation is between -0.03 and 0.03, which include situations where the divergence estimate for one of the 11 splits is order of magnitude higher than for the remaining 10 (662 cases in total). This is also a result of cases, in which the divergence estimator returned constant values for all 11 splits, although the bias varied. Figure 5.13 presents a more detailed breakdown of the 198 dataset/divergence estimator pairs for which this situation has occurred. As it can be seen, the kNN density based Jensen–Shannon's divergence estimator $\tilde{D}_{JS}$ is to blame here, as it was unable to quantify the divergence in the case of 7 out of 26 datasets.

Figure 5.14 depicts the signed bias correlation map averaged over all datasets, while in Figure 5.15 the map averaged over all classifiers is given. These two figures confirm moderate correlation for some combinations of divergence measures and datasets. The blank spots visible in Figure 5.15 reflect the numerical problems of the AMISE Parzen window bandwidth selection



(a) Datasets



(b) Divergence estimators

Figure 5.13: Histograms of datasets and divergence estimators for the 198 constant divergence no–correlation cases (numbers of cases denoted on the vertical axis)

method and kNN density based Jensen–Shannon's divergence estimator mentioned before.

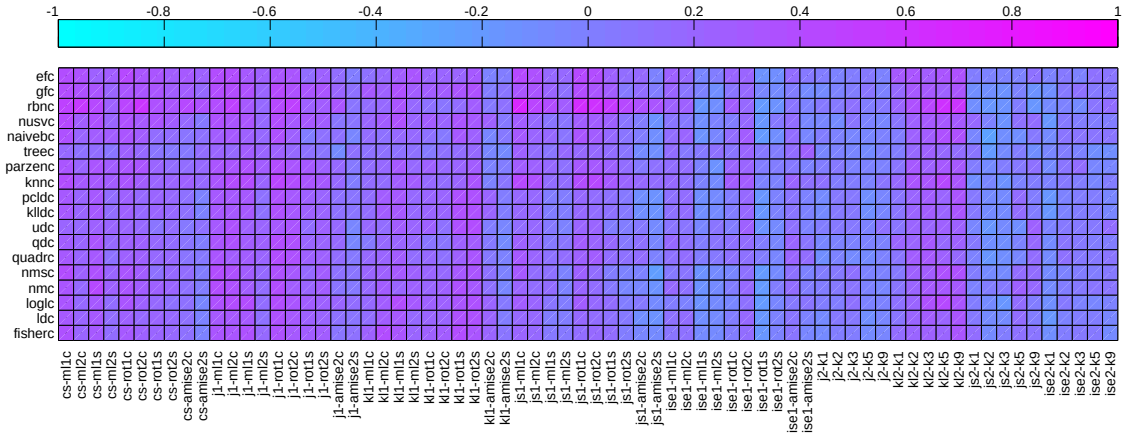


Figure 5.14: Correlation between signed bias and divergence estimates, averaged over datasets

Unfortunately, the averaged results presented so far tend to smooth out the fine details, which might provide more insight into the behaviour of individual methods. For that reason in Figure 5.16 the correlation maps have been given in a breakdown for each dataset. As it can be seen the highest correlation can be observed for the *azi*, *cba*, *chr* and *let* datasets, in all cases for roughly the same divergence estimators (all Parzen window based except for *ise1* as well as the kNN based *kl2*). Unfortunately, for the remaining 22 datasets the situation does not look that

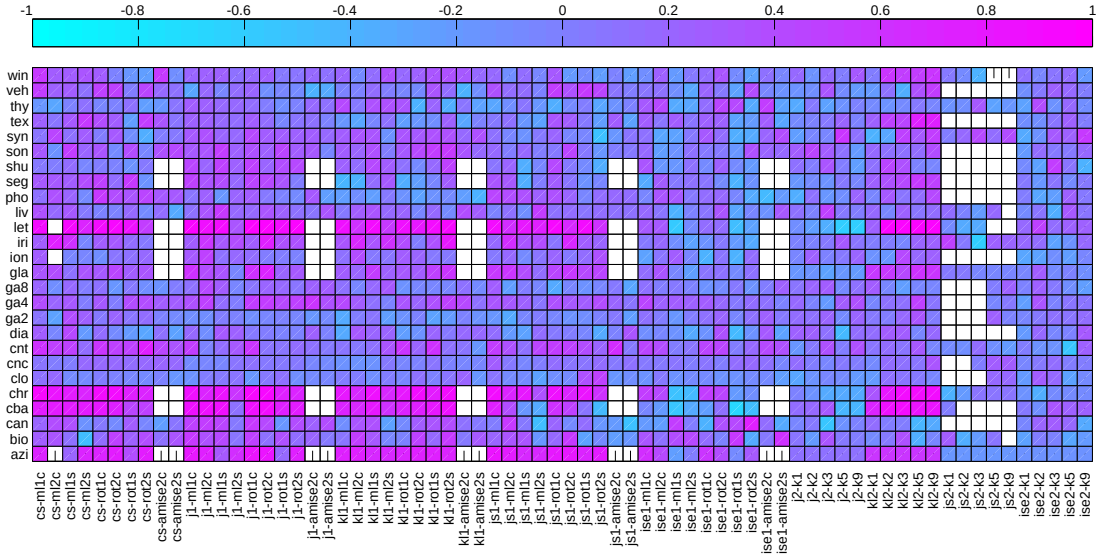


Figure 5.15: Correlation map for signed bias and divergence estimates, averaged over classifiers

well, although for each of them there are areas in the plot denoting medium to strong correlation.

Figure 5.17 presents the histograms of correlation coefficients for individual divergence estimators. As it can be seen, there is only a handful of estimates which demonstrate a certain degree of correlation with the bias, including some of the Cauchy–Schwarz and Kullback–Leibler divergence estimators and especially *kl2-k3*, *kl2-k5* and *kl2-k9*. This seems to contradict the experimental results presented in Figure 5.3, where it can be seen, that the higher the number of neighbours, the slower the convergence of the *kl2* estimators. In the case of *kl2-k1* in Figure 5.17 however, the histogram is symmetric if not skewed to the left, while it changes its shape to more right–skewed as the number of nearest neighbours is increased.

In Figure 5.18 the histograms of datasets, classifiers and divergence estimators for the 806 high (≥ 0.9) signed bias correlation cases have been presented. The first observation is that the correlation is indeed strong only for 3 to 4 datasets and the divergence estimators already identified. The disappointing performance of the *ise1*, *j2*, *js2* and *ise2* estimators has also been confirmed. Also note, that although the histogram of classifiers does not present a uniform distribution, there are numerous high correlation cases for almost all classifiers, with *knnc* and *gfc*/*efc* taking the lead, and *treec* being the worst one.

The most surprising conclusion can be drawn from examination of the four datasets, for which the high correlation has been observed. A closer look at Table A.1 reveals, that the one thing they have in common is a large number of classes, ranging from 20 to 24, while most of the remaining datasets have only 2 to 3 classes. Since in the experimental setting used, the divergences have been approximated for each class in separation, the estimates have been effectively calculated for very small sample sizes (the average class size for the *let* dataset is just 39 instances). From the results of Section 5.4 it is however clear, that for sample sizes of this order the estimates are necessarily far from converging, especially in the case of high–dimensional problems. However, in order to put things into perspective, one needs to realise that the 806 high correlation cases constitute just above 2.6% of the total number of over 30k cases. Thus effectively they form the tail of the distribution depicted in Figure 5.12 and most likely do not have any practical meaning.

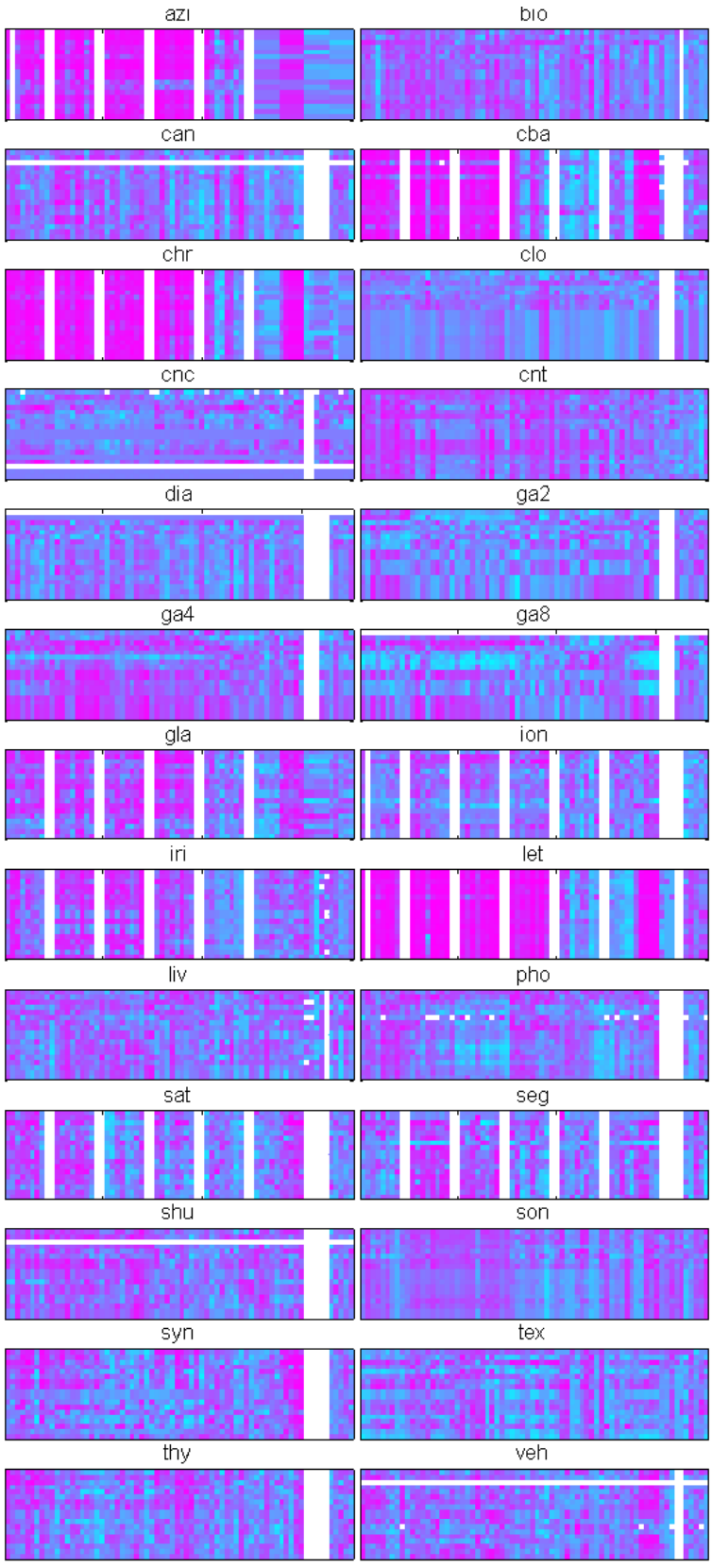
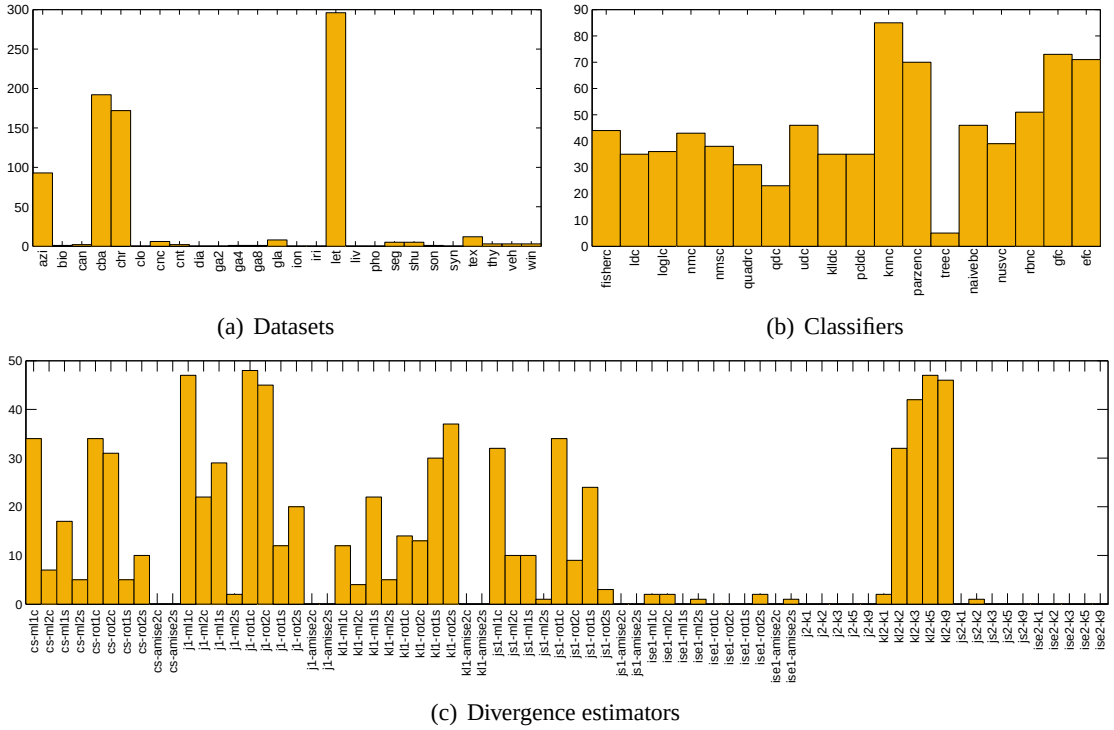


Figure 5.16: Correlation maps for each dataset and signed bias (axes like in Figure 5.14)

Figure 5.17: Correlation coefficient histograms for each divergence estimator and signed bias (X axis: $[-1, +1]$, Y axis: $[0, +40]$)



(a) Datasets (b) Classifiers

(c) Divergence estimators

Figure 5.18: Histograms of datasets, classifiers and divergence estimators for the 806 high (≥ 0.9) signed bias correlation cases (numbers of cases denoted on the vertical axis)

For comparison with the results of Chapter 4 and especially Figures 4.6 and 4.8, scatter plots of the 49 unique subsamples of the Cone–torus dataset (see Section 4.5.1 for more details about the dataset) for the lowest values of all divergence estimators used in the experiments have been depicted in Figure 5.19. The number in round brackets in the title of each plot denotes an identifier of the unique subset. The decision boundaries of a quadratic classifier (*qdc*) have also been superimposed on each plot. The classifier has been chosen due to its stability, so that any drastic changes in the shape of the decision boundaries can be attributed to considerable changes in the structure of the dataset used for training. As it can be seen in the majority of cases, the decision boundaries resemble the ones given in Figure 4.8. The same applies to the banana–shaped class (red circles), which is clearly visible in most cases, similarly to Chapter 4. This can be contrasted to Figure 5.20 containing the scatter plots of 49 unique subsets for the highest values of divergence estimators, where the decision boundaries take on a variety of shapes. As it can be seen though, the properties of the subsamples do depend on the values of the divergence estimators. For the Cone–torus dataset (*cnt*) there was however only a handful of high correlation cases. This behaviour is in fact very similar to that of DPS, where typically 7 out of 8 folds resembled the original dataset when examined visually.

5.6 Discussion

According to the experimental results presented in Section 5.4 it can be said, that in general the divergence between two PDFs is a quantity rather difficult to estimate. This holds regardless of the actual divergence measure chosen, although to a different extent. For example, in the case

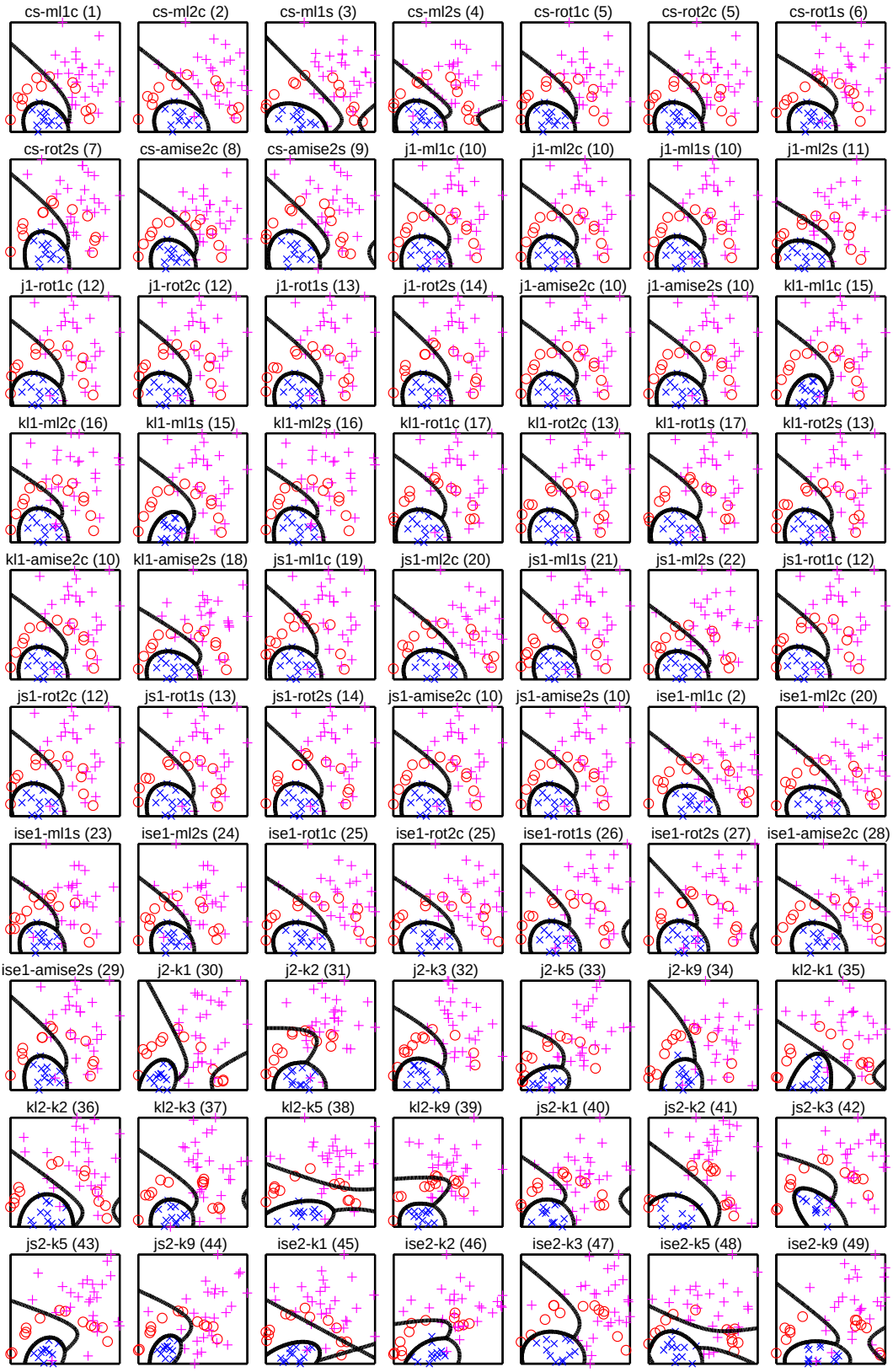


Figure 5.19: Scatter plots of the Cone–torus subsamples for lowest divergence values with superimposed decision boundaries of the *qdc* classifier

Figure 5.20: Scatter plots of the Cone–torus subsamples for highest divergence values with superimposed decision boundaries of the *qdc* classifier

of the Kullback–Leibler divergence, the results are at least disappointing as the estimators often do not reach the true value even for 10000 instances drawn from each distribution. Kullback–Leibler divergence is however one of the most widely used measures of this type.

In [53] the authors use the estimator of Eq. 5.11 to sample a reduced, yet representative subset of the input dataset for the purpose of data condensation. The experimental setting is as follows:

- Experiments based on Gaussian distributions, with (1) three 8–dimensional datasets consisting of two normally distributed classes with various configurations of means and covariances, (2) 100 instances drawn randomly per each class, used for selection of representative subsample and parameter tuning (for each class separately), and (3) 100 instances drawn randomly per each class for testing.
- Experiments based on non–Gaussian distributions, with (1) one dataset of unknown dimensionality having two classes, each distributed according to a mixture of two Gaussians (two–modal distributions), (2) 75 instances drawn randomly per each mode (i.e. 150 instances per class) for selection of representative subsample and parameter tuning (for each class separately), and (3) instances drawn randomly per each class for testing.
- Greedy optimisation of the Kullback–Leibler divergence estimator in both cases.

Although no numerical results are presented, the authors report 'excellent' performance if three or more representatives from each class are selected in the case of the Gaussian datasets and six or more representatives in the non–Gaussian setting. According to the experimental results reported in Section 5.4 for sample size of 100 in most cases it is difficult to expect the $\hat{D}_{KL}$ estimate to approximate the true divergence value well. However, by manipulating the kernel covariance matrix one is able to almost freely influence the value of the estimate. In [53], the authors have set the kernel covariance matrix to be equal to the sample covariance matrix, which led to good performance but only on the Gaussian datasets. This is not surprising as in this case a single kernel function was able to approximate the class PDF well, if located correctly. If one also takes into account that a relatively stable quadratic classifier was used in the experiments the results should be attributed to this specific experimental setting rather than to optimisation of the divergence. The authors admit that 'the selection of the kernel function and the kernel covariance matrix is not clearly understood', which suggests that it is the manual tuning of the covariance matrix which might be responsible for the 'excellent' results in the non–Gaussian scenario.

Surprisingly in most of the literature the Kullback–Leibler or Jeffrey's divergence is not estimated at all. Instead, it is either argued that optimisation of a given objective function is equivalent to optimisation of the divergence between an empirical measure and true yet unknown distribution [109, 120] or closed–form solutions are used restricting the distributions to be Gaussian [61, 112]. Hence it appears that in practice D_{KL} is mostly of theoretical interest, stemming from its connections to Shannon's information theory.

On the contrary, in [159] the authors use an estimator of the Jensen–Shannon divergence in a form similar to Eq. 5.16 but with a different kernel function. In their experimental setting the estimator is calculated for two samples with sizes equal to 1024 and 10240, which according to the results presented in Figure 5.6 is more than enough to obtain accurate estimates, even in a high–dimensional space. Good results reported by the authors are therefore not surprising.

Estimation of the divergence (and other measures for that matter) is hence always connected with a risk resulting from poor accuracy of the estimators, if the datasets do not contain thousands or tens of thousands of instances. This issue is however often neglected by other researchers. An example can be found in [83], where a Cauchy–Schwarz divergence–based clustering

algorithm is derived. The authors report good performance for two synthetic, 2–dimensional datasets (419 instances / 2 clusters and 819 instances / 3 clusters) using the RoT method to determine the 'optimal' Parzen window bandwidth and then heuristically annealing it around that value. These results more or less stay in agreement with the ones presented in Figure 5.10, although they might also be an outcome of manual tweaking of the annealing process.

As for the second experimental part of this study, due to a wide range of datasets used, it can be stated that sampling by optimisation of any divergence measure estimator presented in Section 5.3 does not produce subsets, which lead to consistent observable estimation of the generalisation error. At this stage, it is however difficult to state if the reason for this result is nonsuitability of the divergence measures used or poor accuracy of their estimators, especially in the light of the properties of the datasets for which high correlation has been identified. However due to the strong theoretical foundations of the divergences used, the latter option seems more feasible.

5.7 Concluding remarks

This chapter has been devoted to evaluation of accuracy and empirical convergence of various PDF divergence estimation methods and their application to sampling for the purpose of generalisation error estimation. Five different divergence measures, all having sound theoretical foundations have been examined, paired with two PDF estimators with various parameter settings, leading to 70 divergence estimators in total.

The most important outcome of this study is that the evaluated divergence measures can be estimated accurately only if large amounts of data are available. For some estimators and problems it is possible to obtain accurate estimates for sample sizes in the range of 400–600 instances, while in other cases even 10000 instances is not sufficient. It is important to emphasise here, that empirical convergence has been assessed using simple, synthetic problems with data sampled from Gaussian distributions. Despite this fact, the results are not very encouraging and in fact call into question the practical usability of the examined PDF divergence measures, at least in the situations where their values need to be approximated.

The experimental results of Section 5.4 have however revealed, that although the estimators might be off the true divergence by a large margin, their values nevertheless can differ quite considerably from one fixed size sample to another. Hence there is a possibility, that such estimators can still quantify the similarity between two PDFs, being sufficient for applications in which the relative rather than absolute values of the estimator are of interest. One such application has also been investigated in this chapter.

Building upon the encouraging experimental results of the Density Preserving Sampling technique derived in Chapter 4, the idea was to exploit some of the PDF divergence measures as objective functions of a sampling procedure, in order to obtain a representative subsample of a given dataset, which could be used for accurate estimation of generalisation error. This would in effect further reduce the computational cost of generalisation performance assessment, not only when compared to cross–validation but also with respect to DPS. Unfortunately, in the experiments described in Section 5.5 no strong correlation between the bias and divergence estimators has been identified. Although in some particular cases discussed in previous sections the correlation coefficient exceeded 0.90, these cases account for just above 2.6% of the total number of examined cases and should most likely be credited to a specific set of circumstances rather than to any properties of the divergence estimators used. Hence the PDF divergence measures examined here still remain of theoretical significance, at least from the point of view of generalisation error estimation.

Chapter 6

Applications

6.1 Introduction

In this chapter applications of the techniques developed in this thesis to two real–world environmental problems are discussed.

The first application area is the Quantitative Structure–Activity Relationship (QSAR) modelling and more specifically toxicity level prediction of various chemical compounds, which is a regression problem in high–dimensional sparse space. On this occasion the efforts have been focussed on validating the generalised predictive system development cycle derived in Section 2.2.2 on a real problem, by building a purely data–driven solution able to compete with models developed by the experts in the field. The resultant predictive system has been then submitted to the Environmental Toxicity Prediction Challenge CADASTER 2009[1] and has received an award of the First–Pass Winner. The data provided within the Challenge has also been used in a follow–up study to assess the impact of using the Density Preserving Sampling technique proposed in Chapter 4.

The second application discussed in this chapter is the prediction of water pollution using biomarker data. Due to low quality of the data (i.e. missing values, specific data acquisition process), the obtained predictive system not only takes advantage of the proposed generalised predictive system development cycle, but also tries to exploit the developments of Chapter 3 and serves as another real–world validation study of the DPS technique developed in Chapter 4.

6.2 Ridge regression ensemble for toxicity prediction

6.2.1 Background

Chemical toxicity is a degree of inorganic substances being poisonous and is thus related to various negative biological effects, like gene damage or carcinogenicity. With thousands of new industrial chemicals being synthesized every year and many of them being produced in high volume, the importance of toxicity assessment is evident [160].

Traditional ‘in vivo’ methods of assessing chemical toxicity of various compounds require tests on animals, which not only raises ethical concerns but is also expensive. According to recent estimates, current legislation may lead to a demand for as many as 45 millions laboratory animals in the next 10 years [78]. Computational (‘in silico’) predictions, obtained from a set

[1] http://www.cadaster.eu/node/65

of automatically generated descriptors using only structures of molecules as an input, thus appear as a viable and cost effective alternative. For this reason, Quantitative Structure–Activity Relationship modelling has become an active research area in the recent years [91, 132, 133].

The main issue the researchers are struggling with is that the chemical space has a very high dimensionality. The necessity to deal with even thousands of attributes, the number of which often exceeds the number of instances in the dataset by an order of magnitude or more, is nothing uncommon in QSAR modelling. As a result, any feasible to obtain amount of training data covers only a fraction of the whole input space [160], leading to low, varying and difficult to estimate external predictive power of the models [185].

6.2.2 Environmental Toxicity Prediction Challenge

The predictive system described in the following sections has been developed using the data available within the Environmental Toxicity Prediction Challenge CADASTER 2009, organised by International Conference on Artificial Neural Networks (ICANN'09)[2], European Neural Network Society (ENNS)[3] and the CADASTER project[4]. Since toxicity prediction against animals is a very complex issue, the Challenge has focused on prediction of chemical toxicity against T. pyriformis – a commonly accepted toxicity–screening tool [113, 153] – using previously unpublished dataset. Evaluation of all submitted models have been conducted using the following criteria[5]:

1. Methods with Root Mean Squared Error (RMSE) non–significantly different from the method with lowest RMSE were identified as the First–Pass Winners.
2. Methods providing the best likelihood criteria between estimated and observed confidences for the blind test set were identified amid the First–Pass Winners.

6.2.3 Data description

The data provided in the challenge came in the form of five datasets, each consisting of different attributes (descriptors) and generated directly from the structures of molecules using various approaches and software. Additionally, each dataset was divided into three, non–overlapping parts: training data, known–test data and blind–test data. The measured toxicity values have been given for the two former parts only, and the task was to produce prediction for blind–test data. The details of the data can be found in Table 6.1, and the datasets can be downloaded from the CADASTER website[6]. For the development of the predictive system described here all five datasets have been concatenated, which has resulted in a single set with 2251 attributes. After removing the attributes with constant values, their number dropped to 2048.

6.2.4 Main assumptions

As mentioned before, this study has been perceived as an opportunity to validate the generalised predictive system development cycle described in Section 2.2.2. Hence at this stage the predictive system has been built in a semi–automatic manner, with manual examination of the results of

[2] http://www.kios.org.cy/ICANN09
[3] http://www.e-nns.org/
[4] http://www.cadaster.eu/node/4
[5] http://www.cadaster.eu/node/65
[6] http://www.cadaster.eu/node/67

Table 6.1: Details of toxicity prediction descriptors and training/test datasets

descriptors	# attributes	# train	# known–test	# blind–test
E–state indices[a]	60	644	449	120
DRAGON[b]	1664	644	449	120
SimulationsPlus[c]	221	644	449	120
QuantumChemistry[d]	23	644	449	120
MOE[e]	283	644	449	120
CONCATENATED	2251	644	449	120

[a] E–state indices have been calculated at Virtual Computational Chemistry Laboratory site, http://www.vcclab.org/lab/indexhlp/etstate.html

[b] DRAGON descriptors have been calculated at Virtual Computational Chemistry Laboratory site using optimised structures, http://www.vcclab.org/lab/edragon

[c] SimulationsPlus descriptors have been calculated using ADMET Predictor, http://www.simulations-plus.com/, http://www.cadaster.eu/sites/default/files/challenge/Descriptors_400.pdf

[d] Quantum Chemistry descriptors have been calculated using AM1 MOPAC 7.1 and optimised structures, http://www.vcclab.org/lab/alogps

[e] MOE (Molecular Operating Environment) descriptors have been calculated using optimised structures, http://www.cadaster.eu/sites/default/files/challenge/descr.htm

each step, guiding the choice of methods and models in the following steps. There was also an important constraint on the amount of the available computational resources at that time (a single desktop PC). In this setting the following factors of success have been identified: (1) wide choice of base models, preprocessing and postprocessing methods, (2) computational power sufficient to test many combinations of the above, and (3) experience allowing to take correct decisions or at least informed guesses at each step, in the absence of full information. The importance of the third factor should be stressed here, as it allowed to limit the computational requirements to a large extent.

6.2.5 Generalisation error estimation

Error estimation plays a central role in the generalised predictive system development cycle, as it is used for evaluation of both candidate models (base model/preprocessor pairs) and ensembles. Since at the time of the Environmental Toxicity Prediction Challenge the DPS method derived in Chapter 4 had not been developed yet, the original study used the random subsampling technique (repeated hold–out) and cross–validation for the purpose of generalisation error estimation (please refer to Section 4.2 for details of these methods). However, in Section 6.2.9 the results obtained using 8– and 16–fold DPS–U are also reported for comparison.

6.2.6 Base model pool

The problem of low external predictive power of QSAR models can have different causes. One of them is the possible divergence between distributions underlying the training and testing data, resulting from the immense dimensionality of the chemical space. Unfortunately, there is not much that can be done about it, maybe except from making some assumptions about the unknown–test data distribution, which may or may not be correct. The second possible cause is closely related to model generalisation ability and the problem of overfitting, which can be addressed by controlling the complexity of the final model [11].

In general, complexity of a solution can be controlled by (1) the number of explicit parameters (degrees of freedom) of a model, (2) the number of meaningful attributes the model works with (input dimensionality), or (3) both. The meaningful attributes are understood here as the attributes which carry some information (e.g. ones which are not constant). An interesting example of models susceptible to both these approaches are Artificial Neural Networks [11]. In ANNs the complexity can be controlled either by the number of hidden units/layers or by the dimensionality of the input space, as each additional input increases the number of input weights. There are however methods for which the complexity can be controlled in only one way. For example, in linear regression the number of degrees of freedom depends only on the input dimensionality.

Due to the large number of attributes present in the dataset described in the previous section, the second complexity control mechanism mentioned above seemed more appropriate. The choice of a relatively simple and fast to train base model built using a possibly large number of attributes has been inspired mainly by the theory behind the Support Vector Machines [31, 176]. SVMs are learning systems that use a hypothesis space of linear functions in high–dimensional feature spaces [31]. In such spaces any feasible to obtain amount of data is usually not enough to train a model with many parameters but at the same time even a simple model has enough degrees of freedom to adapt well to the data. However, to allow for greater flexibility, the base model pool consisted of 8 methods listed in Appendix B.2, including a regression boost (*boostr*) technique developed for the purpose of the discussed Challenge. Unfortunately, despite good performance, the *boostr* method had to be abandoned at an early stage of the experiments due to high computational complexity (it required hundreds of base models to be combined). According to initial experiments, ridge regression has proven to be the most promising technique.

Ridge regression

Linear regression is perhaps the best known, most studied and one of the simplest techniques in regression analysis. Its extension into the multiple variable case, the so called 'Multiple Linear Regression', is still being extensively and successfully used in various forms, e.g. in the Partial Least Squares (PLS) or Support Vector Regression methods [56, 176].

A frequent issue in multiple linear regression analysis is the collinearity or near–collinearity of the input variables, leading to ill–posed problems, for which no unique least squares solution exist. In order to circumvent this issue, various techniques have been developed, one of which is the Ridge Regression also known as Tikhonov regularisation [164]. The idea is to introduce an extra regularisation term into the error function. Instead of minimising the usual $\| \boldsymbol{Ax} - \boldsymbol{b} \|^2$ (linear least squares), where $\| \cdot \|$ denotes the Euclidean norm and $\boldsymbol{Ax} = \boldsymbol{b}$ is the overdetermined system of linear equations, the objective function becomes $\| \boldsymbol{Ax} - \boldsymbol{b} \|^2 + \| \boldsymbol{\Gamma x} \|^2$, where $\boldsymbol{\Gamma}$ is the regularisation or Tikhonov matrix, usually chosen to be a multiple of the identity matrix. Ridge regression thus introduces a new parameter, which decides how much the solution departs from the Ordinary Least Squares (OLS) regression and improves the conditioning of the problem. An important feature of ridge regression is that a closed–form solution exists, which makes the regressor fast to train unlike more advanced methods, which require gradient based and often suboptimal optimisation procedure.

For the sake of simplicity, in further experiments the Tikhonov matrix has been set to be equal to the identity matrix.

6.2.7 Data preprocessing

Due to the limited computational resources only two dimensionality reduction techniques have been used: Principal Component Analysis and 'plus–L–takeaway–R' attribute selection method.

Principal Component Analysis

Principal Component Analysis is a standard and commonly used statistical dimensionality reduction technique, often constituting one of the major data preprocessing steps. It is a procedure for linear transformation of a number of possibly correlated variables into a smaller number of uncorrelated variables called principal components (PCs). The first principal component accounts for as much of the variability in the data as possible, and each consecutive component accounts for as much of the remaining variability as possible [40].

The plot of 10–fold cross–validation MSE v. the number of principal components using ridge regression is given in Figure 6.1. As it can be seen, the estimate is overoptimistic between approximately 30 and 100 principal components, that is where the actual minimum of the CV error is located, and for more then about 500 components. Moreover, the CV and test errors diverge in the most interesting range between 150 and 450, where the actual minimum of test set error almost coincides with a maximum of the cross–validation error, with the difference of the MSE reaching up to 100%.

The results obtained are in fact comparable to the ones reported in the literature, both for individual models and their combination (Table 6.2). The minimal known–test error is at the same level as the performance of the best individual model reported in [161, 185], while the known–test error at the minimum of the CV error (for 74 principal components) is about 16% lower then the mean performance of all individual methods described in the papers referred to above. Selecting the final model on the basis of the CV error would thus result in known–test data MSE of 0.2359, ranking the solution in a 3^{rd} place when compared to 11 individual models from [161, 185].

An attempt to improve the results discussed above by using more advanced regression models did not succeed. For example, a cross–trained ensemble of over 100 ANNs with two hidden layers each, developed using 74 principal components has produced MSE equal to 0.2055. While it is better then the performance of ridge regression, the ensemble took incomparably more time to train and yet failed to outperform the best models reported in the literature [161, 185].

Table 6.2: Regression error on PCA transformed dataset / errors reported in the literature

error	**MSE**	**RMSE**
minimal CV (cross–validation)	0.2150	0.4637
known–test at minimum of CV	0.2359	0.4857
minimal known–test	0.1985	0.4455
combined model (literature)	0.1845	0.4300
best individual model (literature)	0.1932	0.4400
mean of all individual models (literature)	0.2745	0.5210
worst individual model (literature)	0.3603	0.6000

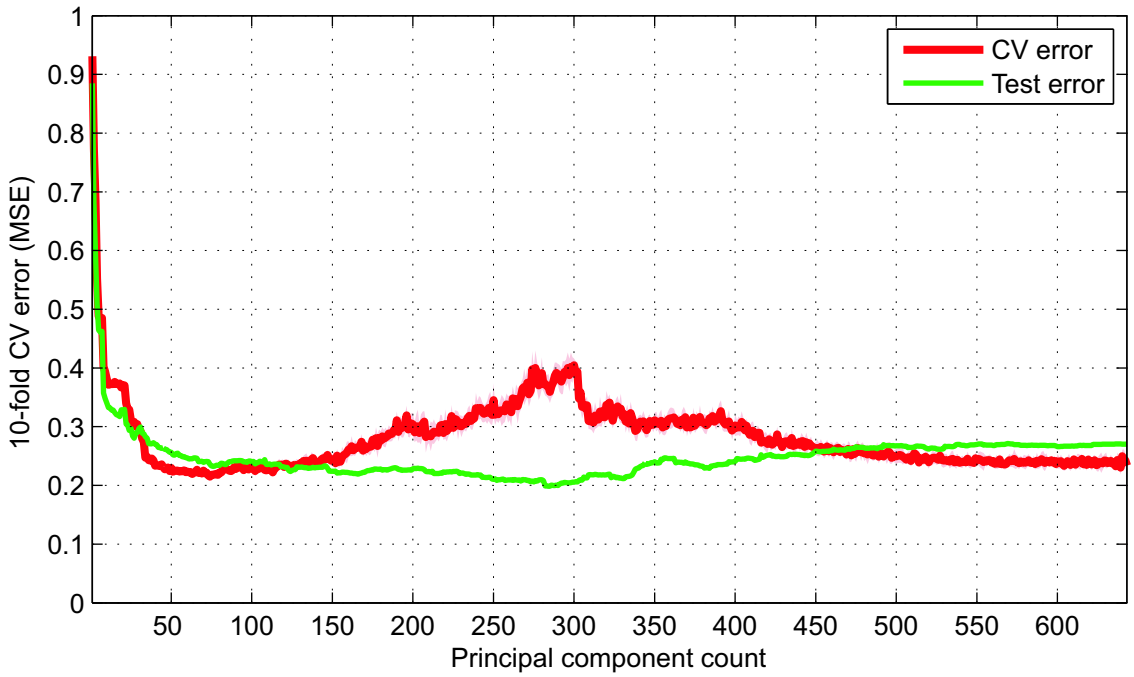


Figure 6.1: Regression error (MSE) on PCA transformed dataset v. the number of principal components

Greedy attribute selection

The only attribute selection method which guarantees finding the global optimum of the evaluation criterion is exhaustive search. Unfortunately, this approach is only feasible for small datasets, as the size of the search space grows exponentially with the number of attributes. For this reason, many approximate search methods have been developed (some examples have been given in Section 2.2.2). Due to the computational limitations only a greedy 'plus–L–takeaway–R' method [175] has been included in the preprocessor pool.

The idea behind greedy methods is to make a locally optimal decision at each step of the optimisation process. For feature selection this can for example mean starting with an empty subset and iteratively adding a single best attribute at each step (forward selection) or starting with the full attribute set and iteratively removing a single worst attribute (backward selection). The 'plus–L–takeaway–R' method is a combination of both forward and backward selection, taking L forward steps followed by R backward steps and repeating the whole procedure until some termination criterion is met.

For the purpose of the Challenge, the following attribute selection procedure has been thus devised. The data has been randomly divided into training (2/3) and hold–out set (1/3). The L and R parameters have been set to 4 and 3 respectively, which was a compromise between search space coverage and algorithm running time. Then, at each step of the feature selection procedure a ridge regressor has been trained using the training set and tested using concatenated training and hold–out sets. If the test error decreased, the new attribute was accepted. The above procedure was repeated 10 times for various random divisions of the data.

In the follow–up study the hold–out method has been replaced with 8–fold DPS–U, producing 8 largely overlapping feature subsets. This has been a manifestation of the phenomena discussed in Section 4.5.2 on the occasion of building combined models based on the DPS folds – the resultant models are very similar and combining them makes little sense.

6.2.8 Ensemble generation and evaluation

As discussed in Section 2.2.2 the ensemble should be built of members which are diverse [99]. One way to encourage diversity is to use different subsets of attributes to train the individual base models. In the discussed case it also appears as the most natural approach, since the feature selection procedure has already produced 10 different, although partly overlapping feature subsets.

Before constructing the final ensemble, yet another step was taken in order to increase the weights of good attributes in the combination. For all subsets of features from each step of each iteration of feature selection, 100 times repeated 10–fold cross–validation was run and all models with error within one standard deviation of the best model's error have been chosen. This has resulted in a total of 29 ridge regressors trained on various subsets of attributes, which were finally combined by averaging, as the postprocessor pool consisted of mean and median combiners.

In the follow–up study, 8– and 16–fold DPS–U has been used instead of cross–validation, considerably reducing the computational requirements and resulting in ensembles of 30 and 27 ridge regressors respectively.

6.2.9 Experiments

The results presented in this section form the submission of the model predictions to the Environmental Toxicity Prediction Challenge CADASTER 2009. The challenge has attracted over 100 participants from 25 countries. Since the contestants were supposed to submit predictions for both known–test and blind–test data, and according to the model development workflow proposed by the organisers[7], two different models have been developed:

1. Intermediate model, utilising 644 instances from the original training set and used to make predictions for the 449 instances from the known–test dataset. Although the known–test instances have not been used to train the intermediate models, the best one in terms of known–test data error out of multiple runs of the algorithm has been chosen for submission.
2. Final model, developed utilising 644+449 instances from the concatenated training and known–test sets and used to make predictions for the 120 instances from the blind–test dataset. The numbers of ensemble members given in previous section apply to the final model.

The challenge results have been given in Table 6.3 and can also be found at the CADASTER project website[8]. The last two rows of the Table present the results for 8– and 16–fold DPS–U used instead of CV during ensemble generation.

10 First–Pass Winners have been chosen with RMSE for the blind–test data non–significantly different from the best model according to the bootstrap test with $p < 0.05$. The predictive system described here was ranked as 6^{th}. Note, that the difference for the blind RMSE between the best and worst method is only 0.054, while the same difference for known RMSE reaches 0.177.

The diagrams with predicted v. target values for the blind–test dataset for the 6 highest ranked methods, including the method described here have been given in Figure 6.2. Notice, that all methods tend to make highest errors on the same, apparently difficult to predict instances, which in fact determines their accuracy.

[7] http://www.cadaster.eu/node/71

[8] http://www.cadaster.eu/final_results.html

Table 6.3: First–Pass Winners ranking / DPS–U performance

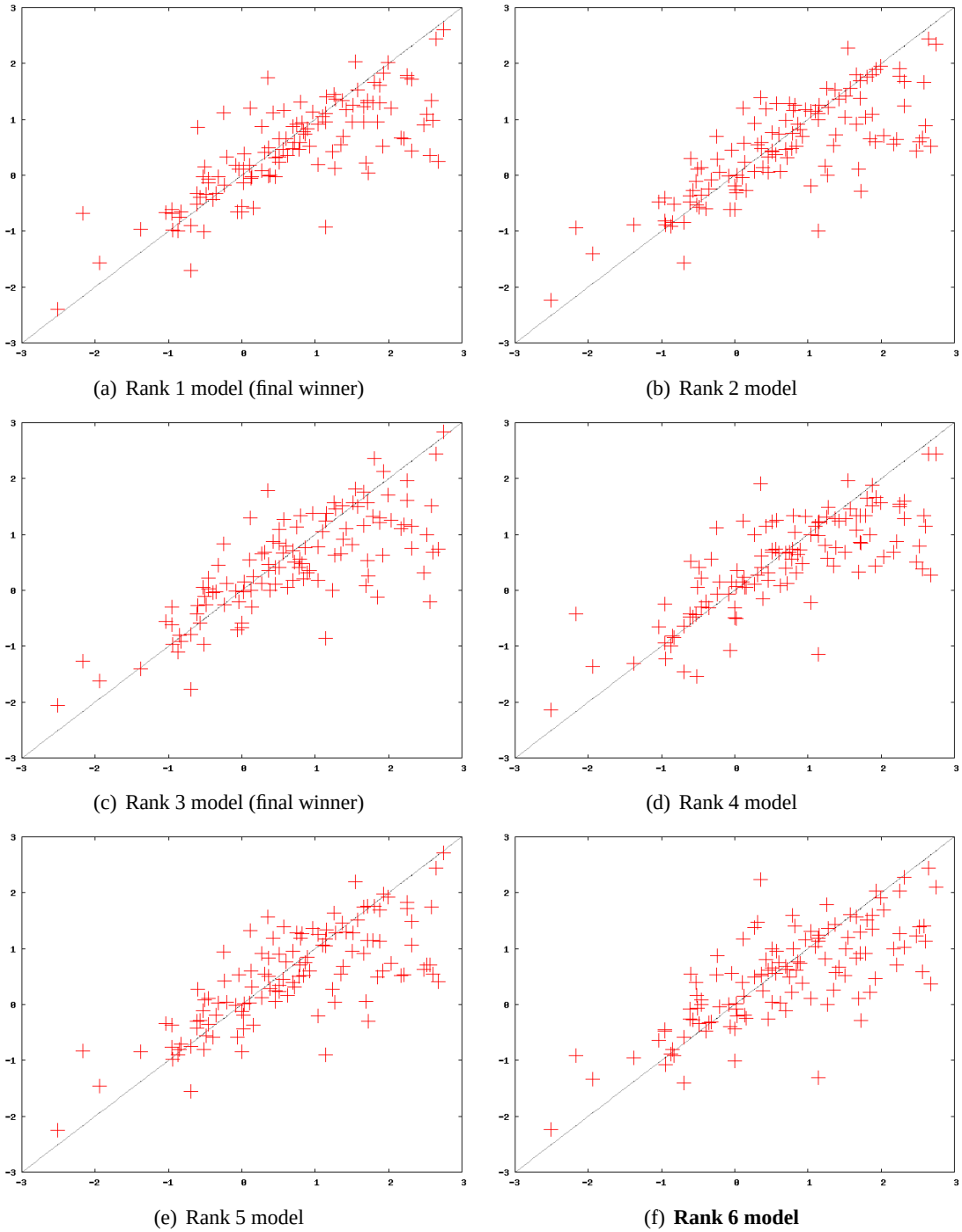
rank	RMSE (blind)	RMSE (known)	RMSE (both)
1	0.741	0.353	0.463
2	0.742	0.367	0.472
3	0.756	0.410	0.503
4	0.760	0.292	0.435
5	0.765	0.395	0.497
6	**0.778**	**0.288**	**0.439**
7	0.789	0.424	0.523
8	0.794	0.416	0.519
9	0.794	0.400	0.509
10	0.795	0.247	0.426
8–fold DPS–U	0.780	0.290	0.441
16–fold DPS–U	0.772	0.279	0.433

6.2.10 Summary of case study findings

The toxicity prediction system described here has been designed by following a rigorous step by step design cycle proposed in this thesis, facilitated and supported by numerical optimisation and comparative evaluations, which was made possible by availability of computational power and advanced data processing algorithms. This in turn has allowed to successfully offset the lack of expert knowledge in the area of toxicity and QSAR modelling by additional computations.

Good performance of the proposed system, evaluated within the Environmental Toxicity Prediction Challenge CADASTER 2009, validates the proposed design cycle and confirms the potential of data–driven approaches for various applications, including but not limited to QSAR modelling. This is an important outcome, as obtaining and integration of expert knowledge into the model is usually expensive, time–consuming and not always possible, while computational resources regularly become cheaper and more accessible. The success of described method also demonstrates that the computational power of modern, easily accessible computers coupled with a rigorous data–driven predictive model development methodology, can be sufficient to develop good solutions without the need for domain expertise, resulting in a more cost effective and faster model development process.

The physically inspired Density Preserving Sampling method has been used in a follow–up study for the purpose of comparison with techniques used during the development of the original predictive system. Although as expected, DPS was not very useful for generation of diverse ensemble members, it has proven well suited for selection of models to be included in the final combination. This has allowed to considerably decrease the computational requirements, resulting in a final system with virtually the same performance as the original one. Hence although the system developed using DPS would still be ranked in the 6^{th} place in the Challenge, the computational savings would allow for e.g. more exhaustive feature subset search or employment of additional regression techniques.



(a) Rank 1 model (final winner)

(b) Rank 2 model

(c) Rank 3 model (final winner)

(d) Rank 4 model

(e) Rank 5 model

(f) **Rank 6 model**

Figure 6.2: Predicted v. target values for the blind–test data

6.3 Predictive modelling of water pollution using biomarker data

6.3.1 Background

Water pollution monitoring becomes a crucial problem as more contaminants enter the marine environment every year [105]. The current trend is prediction of the toxicity level using various measurable attributes of the aquatic environment [118]. This can be observed by a worldwide increase in the number of water quality research funding opportunities by the European Commission[9], the National Research Council in Canada and USA[10,11], as well as various local Councils. The data used in this research has been collected as a part of the 'Marine Environment IQ' project[12] funded by the Research Council of Norway[13], which run between 2006 and 2008.

The condition of a marine environment not always can be diagnosed by chemical analysis of the water, as it does not provide any information on the health of the organisms. Moreover it may also fail to detect any pollution at all due to its low, yet biologically significant degree or very slow increase of contamination level. The solution to this problem is the use of biomarkers.

For many years biomarkers have been successfully used as a tool of exposure analysis. Their importance results from the fact, that they enable detection of pollutants not possible to achieve by other, commonly used methods like chemical or physical analysis [122, 116]. A large number of biomarkers related to their potential effect on organisms has been developed in the literature (for a full list of references please see [21]). Although biomarkers play a significant role in ecotoxicology and environmental risk assessment, they are sometimes difficult to interpret. It is problematic to determine whether a biomarker response is an indicator of impairment or is a part of the homeostatic response, indicating that an organism is successfully dealing with the exposure [49]. When dealing with mixtures of pollutants, a group of biomarkers ('battery') is usually used [25, 43], combining the effect and exposure tests. One of the objectives of this study was to validate the choice of biomarkers made during the 'Marine Environment IQ' project.

The acquisition of biomarker data is an involved process, which requires performing a set of usually destructive tests on biological material. The indicator species of choice are often mussels, which have been used as sentinel organisms from the 1970s [60]. There are multiple advantages of using bivalves in environmental monitoring as they are widely distributed, sedentary and easy to sample, they tolerate a wide range of environmental conditions and bioconcentrate environmental toxicants due to their high filtration activity. Unfortunately, in the majority of studies it is impossible to use the same animal for the whole battery of tests, because of the quantity of biological material required to perform chemical analyses. This dramatically reduces the quality of data by introducing missing attribute values and can have even more serious consequences. It is a common practice to pair the organisms in order to have enough material to perform the chemical tests. This can however change the statistical properties of the data, leading to unexpected behaviour of developed models, including false, highly positively biased accuracy estimates, which in consequence renders the models useless.

After the data has been collected it can be processed, which is the main focus of this study. Although there have been several approaches to water quality prediction in the literature using ANNs [108], self organising maps [3], Bayes networks [131] and other methods [66], none

[9] European Commission Research, http://ec.europa.eu/research/index.cfm

[10] National Research Council Canada, http://www.nrc-cnrc.gc.ca/eng/index.html

[11] National Research Council, http://sites.nationalacademies.org/NRC/index.htm

[12] Developing an Index of the Quality of the Marine Environment (Marine Environment IQ) based on biomarkers: Integration of pollutant effects on marine organisms, http://www.iris.no/Internet/NFR-feb2009.nsf/

[13] Research Council of Norway, http://www.forskningsradet.no/

of them was using biomarker data. From the point of view of data modelling, the biomarker data usually has low quality due to the missing values, high dimensionality and small size of the dataset, which can cause various issues [11, 40]. Perhaps the most important of them is to define what does one expect the data to reveal and is the data adequate for this purpose.

6.3.2 Dataset properties

The dataset contains a collection of biomarker data measured on mussels at 4 different marine stations located in South–West Norway (Rogaland County), in the course of a 4–week experiment. The locations of the sites can be seen in Figure 6.3. The stations have been chosen according to known water pollution levels (see references within [21]) and the goal of the study was to provide field data to investigate possible biomarker combinations to discriminate between various pollution levels. There are 50 instances in the dataset, each having 12 attributes. There are also 5 different classes, denoting the 5 stations, and 4% of attributes are missing. The details of the classes have been given in Table 6.4. For the list of biomarkers used see [21].

Table 6.4: Mussel biomarker data class details

class	# instances	description	missing values
T0-C	10	Control (clean) site at experiment start	10.0% (12 of 120)
T4-C	10	Control (clean) site after 4 weeks	0.0% (0 of 120)
T4-S1	10	Lightly polluted site after 4 weeks	2.5% (3 of 120)
T4-S2	10	Moderately polluted site after 4 weeks	2.5% (3 of 120)
T4-S3	10	Heavily polluted site after 4 weeks	5.0% (6 of 120)

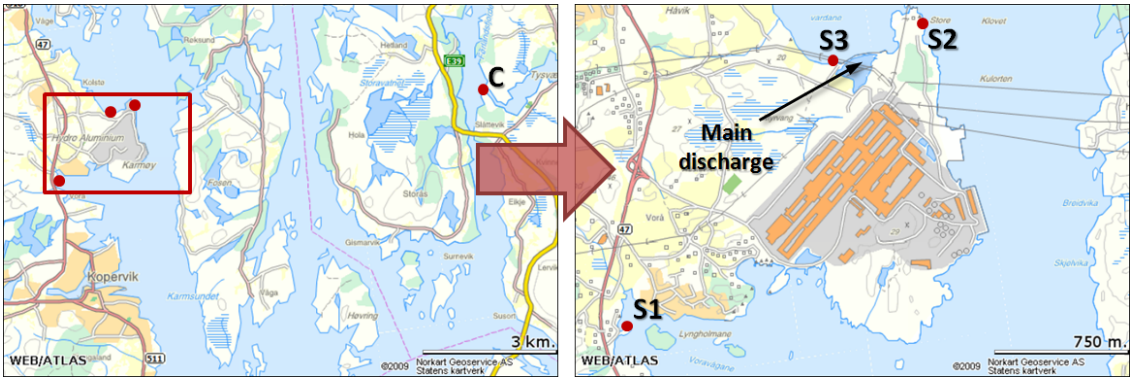


Figure 6.3: Locations of the marine stations[a]

[a]Maps courtesy of WebAtlas: http://www.webatlas.no/

From the point of view of building a usable classification model, a number of difficulties can already be expected even before examining the data in detail. The difficulties are:

1. Small dataset size. This results in the lack of ability to use more advanced models with many degrees of freedom/parameters (e.g. all but the smallest neural networks) [126] and negatively influences the reliability of the model generalisation ability estimate.
2. Relatively high dimensionality of data, which is higher than the number of objects per class and can pose a whole number of difficulties (‘curse of dimensionality’ [11]).

3. Missing attributes. Although the missingness level is low, most machine learning techniques do not natively support incomplete data. Moreover, from the statistical standpoint, the mechanism behind missingness is not known. The only information is that the data is missing due to some biological tests going wrong in one way or another, but it is not known if there exists any relation between the values of measured parameters and the test going wrong. A common simplifying Missing At Random [136] assumption may thus not hold and some form of a missingness model [117] may be required.

In order to gain some insight into the structure of the dataset basic statistical analysis has been performed, which corresponds to the 'data inspection' step of the generalised predictive system development cycle derived in Section 2.2.2. For the estimation of statistical properties of the data, the missing values have been temporarily ignored and the dataset has been scaled to fit into the $[0, +1]$ range.

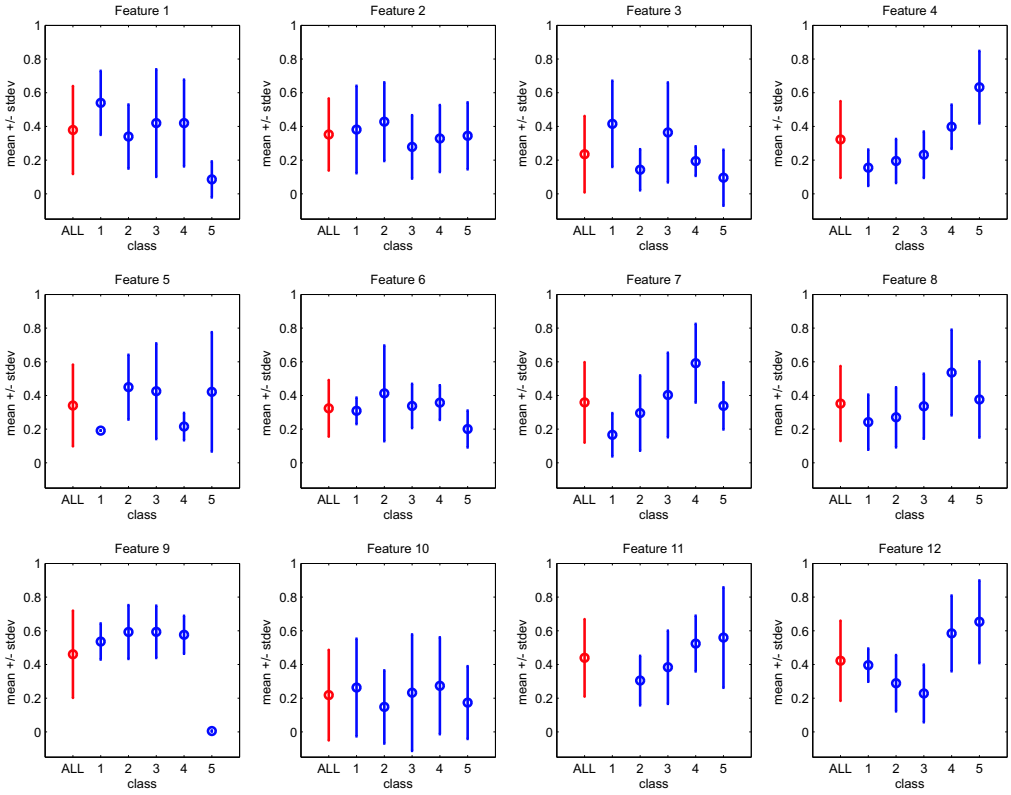
1. **Mean and standard deviation.** The mean and standard deviation values for all attributes have been depicted in Figure 6.4(a), with the leftmost bar representing the whole dataset and remaining bars representing classes T0-C to T4-S3, left to right. As it can be seen, for example feature 9 alone may facilitate distinction between the most heavily polluted site and all the others. Note also, that features 1, 3 and especially 5[14] might as well be used to discriminate between classes T0-C and T4-C – the control site at the beginning and end of the experiment. This suggests some additional dependency in the system, as the feature values at the control site change over time although the pollution level does not. This phenomenon, known as concept drift [172], may render the predictions of the model less accurate as the time passes by.
2. **Probability density functions.** Class conditional probability density functions for each of the features are given in Figure 6.4(b). In almost every case the distributions overlap, thus none of the features alone is sufficient to discriminate between the classes. The exception is feature 9 – the class representing the heavily polluted site appears well separated. Also the peaks of the distributions of feature 11 form two, at least partially separated groups.

6.3.3 Classification with a single model

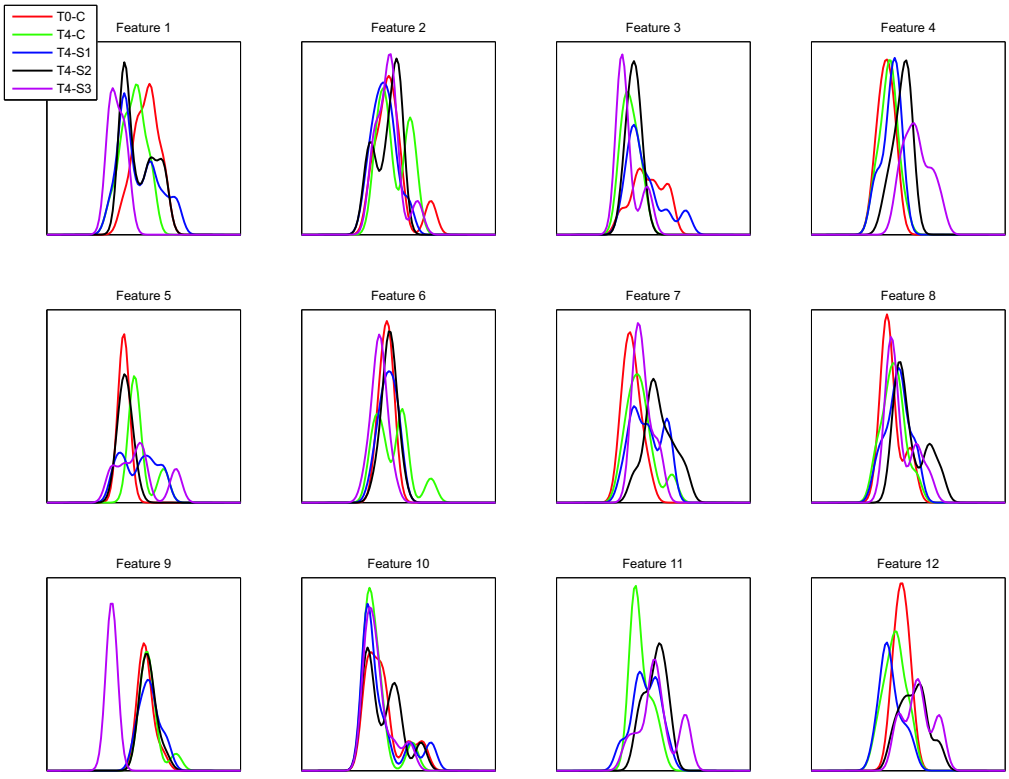
In order to quickly obtain a number of working prototype models, a simple experiment using a set of standard classifiers has been designed. It not only allowed to get more detailed insight into the dataset and confirm the difficulties listed in Section 6.3.2, but also provided first performance estimates. Most importantly however, this initial analysis has helped to define the classification problem, as the provider of the data wasn't entirely sure what can be done with it.

As in the previous chapters the classifiers used in the experiments are a part of the PRTools [42] toolbox (see Table B.1 for details). The original experiments have been primarily performed within a repeated 10–fold cross–validation scheme. Due to small size of the dataset and in order to obtain a better picture of possible performance, some experiments have been rerun using leave–one–out cross–validation. In the follow–up study the supervised Density Preserving

[14] An unexpected behaviour of some of the models has been observed during the feature selection experiments. A simple Nearest Neighbour classifier trained on feature 5 alone produced a 0% 10–fold and 0% leave–one–out cross–validation error and the leave–one–out error of *qdc* in scenario 1 was also suspiciously low – 6%. It has turned out to be a result of pairing the organisms to have enough material to perform the chemical tests. For this reason, feature 5 has not been used in further analysis.



(a) Mean values and standard deviations of all features for each class



(b) Class conditional probability density functions (colours denote classes)

Figure 6.4: Statistical properties of the data

Sampling (DPS–S) technique of Chapter 4 has also been used for comparison purposes. Since the DPS method has not been designed with missing data in mind, the Local Dimensionality Reduction technique proposed in Section 3.5.1 has been used here, enabling calculation of distances between deficient data instances.

As the percentage of missing features was rather small, at this stage a simple class–conditional mean imputation approach discussed in Section 3.4.2 has been used to fill in the blanks. As in the case of class T0-C all values of feature 11 were missing, they have been replaced with a global mean value for the whole dataset. The dataset has been scaled to fit within the $[0, +1]$ interval, as the ranges of the original features vary greatly and feature 5 has been removed.

Eight experiment scenarios summarised in Table 6.5 have been devised. The goal was to see if any of the scenarios can be ruled–out at the early stage of experiments due to lack of discriminative power of the feature set. Some of the scenarios were chosen only to verify the anticipated difficulties. The results of preliminary experiments can be found in Tables 6.6 to 6.8.

Table 6.5: Experiment scenarios

scenario	**class count**	**class details**
1	5	T0-C — T4-C — T4-S1 — T4-S2 — T4-S3
2	4	T0-C+T4-C — T4-S1 — T4-S2 — T4-S3
3	2	T0-C+T4-C — T4-S1+T4-S2+T4-S3
4	2	T0-C — T4-S1+T4-S2+T4-S3
5	2	T4-C — T4-S1+T4-S2+T4-S3
6	4	T0-C — T4-S1 — T4-S2 — T4-S3
7	4	T4-C — T4-S1 — T4-S2 — T4-S3
8	2	T0-C — T4-C

Table 6.6: 10–fold cross–validation errors

classifier	**scenario**							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
fisherc	0.356	0.379	0.225	0.215	0.197	0.292	0.260	0.415
ldc	**0.310**	0.329	0.208	0.217	0.227	0.273	0.258	0.415
loglc	0.432	0.358	0.202	0.287	0.235	0.400	0.325	0.340
nmc	0.360	0.361	**0.147**	0.178	**0.168**	0.228	0.283	0.300
nmsc	0.356	**0.291**	0.157	0.162	0.200	**0.225**	**0.240**	0.405
quadrc	0.560	0.540	0.338	0.222	0.367	0.488	0.480	0.405
qdc	0.600	0.537	0.338	0.387	0.375	0.505	0.517	0.315
udc	0.564	0.390	0.172	0.318	0.252	0.478	0.320	0.445
klldc	**0.310**	0.329	0.208	0.217	0.227	0.273	0.258	0.415
pcldc	**0.310**	0.329	0.208	0.217	0.227	0.273	0.258	0.415
knnc	0.448	0.450	0.212	0.198	0.440	0.360	0.420	0.375
parzenc	0.422	0.394	0.214	**0.148**	0.282	0.320	0.398	0.335
treec	0.666	0.671	0.327	0.257	0.495	0.558	0.577	0.405
naivebc	0.444	0.440	0.298	0.167	0.335	0.387	0.453	**0.115**
svc	0.388	0.343	**0.147**	0.362	0.505	0.273	0.290	0.315
nusvc	0.374	0.309	0.234	0.265	0.262	0.270	0.248	0.310
mean	0.431	0.403	0.227	0.238	0.299	0.350	0.349	0.358
efc-mimp/ldr	0.360	0.584	0.251	0.197	0.248	0.653	0.810	0.230

Table 6.7: 8–fold Density Preserving Sampling errors (DPS–S)

classifier	scenario							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
fisherc	0.360	0.388	0.150	0.233	0.200	0.325	0.325	0.350
ldc	0.360	0.350	0.167	0.183	0.183	0.250	0.275	0.350
loglc	0.440	0.438	0.217	0.250	0.183	0.375	0.325	0.350
nmc	0.260	0.363	0.142	0.167	0.167	**0.200**	**0.225**	0.200
nmsc	0.300	0.338	0.158	**0.117**	**0.150**	0.225	0.325	0.300
quadrc	0.600	0.525	0.342	0.233	0.333	0.450	0.525	0.300
qdc	0.500	0.475	0.342	0.400	0.383	0.425	0.500	0.250
udc	0.540	0.375	0.217	0.300	0.250	0.475	0.325	0.450
klldc	0.360	0.350	0.167	0.183	0.183	0.250	0.275	0.350
pcldc	0.360	0.350	0.167	0.183	0.183	0.250	0.275	0.350
knnc	0.460	0.388	0.225	**0.117**	0.350	0.300	0.450	0.300
parzenc	0.400	0.388	0.192	0.133	0.283	0.325	0.400	0.350
treec	0.800	0.600	0.392	0.267	0.517	0.475	0.525	0.550
naivebc	0.440	0.475	0.358	0.200	0.267	0.475	0.450	**0.100**
svc	0.400	0.338	**0.108**	0.400	0.467	0.400	0.300	0.500
nusvc	**0.220**	**0.300**	0.133	0.250	0.267	0.225	0.275	0.350
mean	0.425	0.402	0.217	0.226	0.273	0.339	0.361	0.337
efc-mimp/ldr	0.480	0.738	0.208	0.233	0.283	0.725	0.800	0.350

Table 6.8: Leave–one–out cross–validation errors

classifier	scenario							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
fisherc	0.400	0.400	0.267	0.233	0.183	0.300	0.300	0.350
ldc	0.300	0.338	0.183	0.200	0.233	0.225	0.250	0.350
loglc	0.460	0.350	0.192	0.250	0.233	0.400	0.300	0.250
nmc	0.300	0.338	0.142	0.167	**0.167**	0.225	0.250	0.250
nmsc	0.360	**0.313**	0.158	0.167	**0.167**	0.225	0.250	0.400
quadrc	0.380	0.363	0.383	0.233	0.233	0.300	0.375	0.350
qdc	**0.120**	0.350	0.383	0.217	0.183	**0.100**	**0.100**	**0.100**
udc	0.580	0.400	0.158	0.317	0.250	0.500	0.350	0.450
klldc	0.300	0.338	0.183	0.200	0.233	0.225	0.250	0.350
pcldc	0.300	0.338	0.183	0.200	0.233	0.225	0.250	0.350
knnc	0.460	0.475	0.208	**0.150**	0.417	0.400	0.425	0.400
parzenc	0.400	0.400	0.225	**0.150**	0.317	0.325	0.400	0.350
treec	0.660	0.625	0.442	0.233	0.517	0.550	0.500	0.250
naivebc	0.420	0.438	0.292	**0.150**	0.333	0.350	0.500	**0.100**
svc	0.480	0.363	**0.133**	0.400	0.517	0.300	0.325	0.400
nusvc	0.380	0.350	0.217	0.267	0.283	0.275	0.250	0.350
mean	0.394	0.386	0.234	0.221	0.281	0.308	0.317	0.316
efc-mimp/ldr	0.440	0.580	0.233	0.233	0.250	0.675	0.775	0.300

Scenario 1 – all 5 classes

The first experiment involved classification of instances into one of 5 classes from Table 6.4. The mean 10–fold CV error of all classifiers (0.431) is higher than the mean leave–one–out error (0.394) mostly due to suspiciously good performance of *qdc* in the latter case (0.120). Note, that before removing feature 5 from the dataset, leave–one–out *qdc* error was equal to 0.060, while the remaining classifiers performed at a similar level. Note good performance of the Electrostatic Field Classifier in the 10–fold case (classification error well beyond average) and a disappointing result for the leave–one–out method (classification error higher than average).

Examination of the DPS errors from Table 6.7 confirms high classification error of the EFC approach. An important observation is, that although the classifier rankings according to 10–fold CV and 8–fold DPS differ, all top models according to the CV error are in top 3 models according to the DPS error, and that in both cases *treec* is the worst performing classifier.

Scenario 2 – control site and various pollution degrees

For this experiment classes T0-C and T4-C have been combined together to form a single, control class. The results of both cross–validation approaches are once again consistent but not remarkable, although the 10–fold CV mean error of all classifiers has been slightly reduced from roughly 0.430 to about 0.400. The same applies to the mean DPS error over all classifiers. Combination of the two control classes thus seems to have positive influence on the classification error. Moreover, this approach is the only way to address the concept drift issue with this limited amount of data. As a result scenario 2 has been chosen for further experiments.

Note, that this time the top model according to DPS (*nusvc*) is second best according to 10–fold CV and vice versa, and that *treec* is once again the worst classifier in both cases.

As for the EFC approach, the performance is comparable to that of *treec*, thus despite its built–in missing data handling algorithms, EFC is unable to handle this experiment scenario.

Scenario 3 – clean and polluted environment

In scenario 3 classes T0-C and T4-C have been combined together to form a single, control class. Classes T4-S1, T4-S2 and T4-S3 have also been combined to form a single class representing polluted sites. This resulted in a dramatic improvement of the classification accuracy (roughly 0.220 – 0.230 mean error of all three error estimators, 0.147 10–fold CV error of best classifiers (*nmc*, *svc*) and 0.108 DPS error of *svc*). In this experiment scenario the EFC performs at the average level, which compared to the errors of the best classifiers is not a good result.

Scenario 4 and 5 – clean (T0-C / T4-C) and polluted environment

In these two scenarios, classes T4-S1, T4-S2 and T4-S3 have been combined to form a single class representing polluted environment. One of the clean environment classes has then been dropped (T0-C for scenario 4 and T4-C for scenario 5 respectively), effectively reducing the number of classes to 2.

Note the performance gap between those two scenarios (mean 10–fold CV errors), reaching 0.06 in favor of scenario 4. This confirms the presence of concept drift in the data as the discrimination between control site at the beginning of the 4–week experiment and polluted sites is easier. Similar conclusions can be drawn from examination of the DPS error estimates.

Scenario 6 and 7 – control site (T0-C / T4-C) and various pollution degrees

Similarly to scenarios 4 and 5, classes T0-C (scenario 6) and T4-C (scenario 7) have been dropped respectively, while the classes representing various degrees of pollution have remained unchanged. The mean leave–one–out errors for both scenarios are lower than the 10–fold CV errors due to surprisingly good performance of the *qdc* – this issue has been already discussed, so only the latter errors are meaningful in this case.

Scenario 8 – control site at time T0-C and T4-C

This experiment scenario was designed to check if classes T0-C and T4-C are indeed different, so a dataset consisting of objects from these two classes only has been used. Although all three mean errors are quite high (over 0.030), the best performing classifier *naivebc* has produced only 0.115 10–fold CV and 0.100 DPS error. Notice, that no anomalies similar to *qdc* have ever occurred in the case of this particular classifier, so there is no reason to treat its error estimate as unreliable. This confirms that the objects collected at the same site in two different moments have distinct properties, influenced by factors other than the pollution level.

Remarks on the behaviour of EFC and DPS

In the majority of experiment scenarios discussed above, the Electrostatic Field Classifier either performs at an average or better than average level. However, in the most interesting scenario 2, which is the main focus of further experiments, EFC fails. For that reason it has not been included in the base model pool for development of the final predictive system.

The averaged DPS error estimates are very similar to that produced by 10–fold CV. Specifically, in the case of scenario 2, these values are 0.402 and 0.403 respectively. Moreover both methods produce similar rankings of the classifiers, e.g. for the scenario of interest top 2 as well as the worst performing model are exactly the same according to both 10–fold CV and DPS. This suggests that replacement of cross–validation with DPS in further experiments should not negatively influence the performance of the final ensemble model.

6.3.4 Feature selection

As mentioned in Section 6.3.2, the dimensionality of the dataset is relatively high. This can often be very problematic for various machine learning techniques, since they are forced to operate in a sparse space and cannot be trained properly. As a result, reduction of the number of attributes usually has a positive influence on the classification performance.

There is another practical reason for using as few attributes as possible – data acquisition cost. By identifying attributes which are correlated or otherwise irrelevant, the number of required biological tests can be reduced. This not only saves money but also circumvents the issue of limited amount of biological material, which is substantial for the mutually exclusive or destructive tests.

Experiments described in this section aim to investigate which of the features have the lowest discriminative power and how their removal might affect the classification performance. As mentioned before, the experiments were run only for scenario 2 from Table 6.5.

Removal of one feature at a time

The classification results for removal of one feature at the time, thus using 10 remaining features, have been given in Tables 6.9 and 6.10. A modest improvement of both CV and DPS mean classification errors over previous experiments has been observed for removal of features 1, 10 and 11. Further removal of features could possibly improve the results even more, but enumeration of all feature pairs, triplets etc. is a problem of exponential complexity, thus these experiments were not run here. It is clear however that some form of feature selection would improve performance.

Table 6.9: Classification errors after removal of a single feature (10–fold CV)

classifier	feature										
	1	**2**	**3**	**4**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
fisherc	0.360	0.369	0.406	0.381	0.366	0.400	0.404	0.440	0.324	0.348	**0.356**
ldc	0.305	0.376	0.329	0.388	0.323	**0.335**	**0.296**	0.436	0.281	0.318	0.390
loglc	0.454	0.373	0.443	**0.291**	0.309	0.351	0.401	0.435	0.360	0.368	0.393
nmc	0.340	0.350	0.345	0.416	0.348	0.360	0.349	0.446	0.298	0.313	0.380
nmsc	0.284	**0.300**	**0.299**	0.386	**0.266**	0.350	0.324	**0.395**	0.287	**0.261**	0.428
quadrc	0.494	0.502	0.478	0.529	0.481	0.545	0.491	0.559	0.488	0.503	0.511
qdc	0.548	0.501	0.605	0.579	0.559	0.489	0.598	0.603	0.613	0.583	0.574
udc	0.378	0.385	0.400	0.436	0.328	0.436	0.366	0.463	0.345	0.392	0.461
klldc	0.305	0.376	0.329	0.388	0.323	**0.335**	**0.296**	0.436	0.281	0.318	0.390
pcldc	0.305	0.376	0.329	0.388	0.323	**0.335**	**0.296**	0.436	0.281	0.318	0.390
knnc	0.411	0.386	0.439	0.425	0.385	0.476	0.433	0.506	0.381	0.389	0.413
parzenc	0.349	0.361	0.408	0.380	0.388	0.444	0.419	0.497	0.395	0.386	0.401
treec	0.616	0.628	0.663	0.679	0.658	0.663	0.635	0.673	0.659	0.663	0.655
naivebc	0.480	0.457	0.448	0.521	0.459	0.443	0.456	0.546	0.453	0.443	0.447
svc	0.364	0.353	0.350	0.355	0.352	0.404	0.380	0.433	0.326	0.357	0.360
nusvc	**0.281**	0.316	0.368	0.364	0.304	0.351	0.351	0.443	**0.280**	0.326	0.361
mean	0.392	0.401	0.415	0.432	0.386	0.420	0.406	0.484	0.378	0.393	0.432

Table 6.10: Classification errors after removal of a single feature (8–fold DPS–S)

classifier	feature										
	1	**2**	**3**	**4**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
fisherc	0.338	0.350	0.338	0.362	0.387	0.412	0.388	0.475	0.325	0.338	0.350
ldc	0.300	0.388	0.300	0.313	0.350	0.362	**0.300**	0.412	0.275	0.287	0.388
loglc	0.488	0.425	0.375	**0.225**	0.350	0.362	0.438	**0.313**	0.350	0.362	0.413
nmc	0.313	0.350	0.300	0.388	0.400	0.350	0.363	0.400	**0.237**	0.262	**0.287**
nmsc	**0.287**	0.350	**0.275**	0.363	**0.300**	0.362	0.325	0.438	**0.237**	**0.237**	0.388
quadrc	0.438	0.650	0.512	0.538	0.563	0.538	0.375	0.488	0.613	0.550	0.388
qdc	0.550	0.500	0.588	0.500	0.600	0.550	0.550	0.575	0.500	0.550	0.512
udc	0.375	0.375	0.350	0.463	0.337	0.388	0.375	0.412	0.325	0.362	0.512
klldc	0.300	0.388	0.300	0.313	0.350	0.362	**0.300**	0.412	0.275	0.287	0.388
pcldc	0.300	0.388	0.300	0.313	0.350	0.362	**0.300**	0.412	0.275	0.287	0.388
knnc	0.400	0.438	0.388	0.438	0.425	0.425	0.388	0.475	0.463	0.412	0.412
parzenc	0.363	0.362	0.412	0.400	0.412	0.450	0.438	0.425	0.388	0.412	0.412
treec	0.550	0.637	0.675	0.650	0.700	0.575	0.600	0.713	0.600	0.563	0.675
naivebc	0.425	0.375	0.500	0.512	0.525	0.475	0.425	0.613	0.487	0.412	0.438
svc	0.375	0.375	0.362	0.388	0.363	0.400	0.412	0.463	0.338	0.375	0.388
nusvc	0.313	**0.263**	0.350	0.350	0.350	**0.337**	0.400	0.362	0.287	0.250	0.363
mean	0.382	0.413	0.395	0.407	0.423	0.420	0.398	0.462	0.373	0.372	0.419

Classification using a single feature

In this experiment performance of a classifier built on a single feature has been tested. The results are given in Tables 6.11 and 6.12. Non–surprisingly, none of the features alone facilitates acceptable classification performance but there are two features which produce the lowest error (4 and 9). The latter is especially interesting since the PDF plot (Figure 6.4(b)) suggested possible discriminative power to separate class T4-S3 from the remaining ones. An experiment using only those 2 features revealed 0.378 mean CV and 0.345 mean DPS classification errors, which already is an improvement over the results obtained using the whole feature set. This experiment has also uncovered unexpected properties of feature 5 mentioned before.

Table 6.11: Single feature classification errors (10–fold CV)

classifier	feature											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
fisherc	0.663	0.735	0.654	0.575	0.638	0.695	0.612	0.688	0.500	0.786	0.713	0.637
ldc	0.734	0.679	0.573	0.471	0.638	0.678	0.600	0.666	0.537	0.800	0.792	0.574
loglc	0.711	**0.665**	0.613	0.545	0.638	0.650	0.611	**0.643**	0.548	0.807	0.768	**0.531**
nmc	0.734	0.679	0.573	0.471	0.684	0.678	0.600	0.666	0.537	0.800	0.792	0.574
nmsc	0.734	0.679	0.573	0.471	0.638	0.678	0.600	0.666	0.537	0.800	0.792	0.574
quadrc	0.645	0.739	0.570	0.530	0.642	0.595	**0.553**	0.682	0.524	0.812	0.711	0.610
qdc	**0.648**	0.758	0.582	**0.456**	0.714	**0.590**	0.560	0.675	0.541	0.773	0.745	0.659
udc	**0.648**	0.758	0.582	**0.456**	0.714	**0.590**	0.560	0.675	0.541	0.773	0.745	0.659
klldc	0.730	0.679	0.573	0.471	0.638	0.678	0.600	0.666	0.537	0.800	0.792	0.574
pcldc	0.730	0.679	0.573	0.471	0.638	0.678	0.600	0.666	0.537	0.800	0.792	0.574
knnc	0.676	0.779	0.688	0.521	**0.000**	0.793	0.651	0.709	0.533	**0.738**	0.690	0.694
parzenc	0.676	0.713	0.614	0.504	0.548	0.622	0.571	0.778	0.549	0.805	0.700	0.679
treec	0.691	0.745	0.719	0.519	0.340	0.776	0.659	0.656	0.511	0.751	0.764	0.690
naivebc	0.693	0.754	0.710	0.482	0.492	0.596	0.581	0.734	**0.440**	0.832	0.764	0.666
svc	0.749	0.750	0.750	0.573	0.600	0.750	0.676	0.748	0.500	0.750	0.729	0.750
nusvc	0.659	0.838	**0.560**	0.553	0.660	0.760	0.641	0.759	0.506	0.771	**0.636**	0.639
mean	0.695	0.727	0.619	0.504	0.576	0.675	0.605	0.692	0.524	0.787	0.745	0.630

Table 6.12: Single feature classification errors (8–fold DPS–S)

classifier	feature											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
fisherc	0.663	0.725	0.637	0.575	0.725	0.738	0.588	0.688	0.500	0.787	0.762	0.775
ldc	0.750	0.725	**0.563**	**0.450**	0.625	**0.575**	**0.512**	0.650	0.600	0.750	0.762	**0.550**
loglc	0.700	0.675	0.625	0.550	0.650	0.650	0.613	0.625	0.550	0.738	0.737	0.588
nmc	0.738	0.725	**0.563**	**0.450**	0.625	**0.575**	**0.512**	0.650	0.600	0.750	0.762	**0.550**
nmsc	0.750	0.725	**0.563**	**0.450**	0.625	**0.575**	**0.512**	0.650	0.600	0.750	0.762	**0.550**
quadrc	**0.637**	0.775	0.613	0.512	0.600	0.613	0.588	0.688	0.550	**0.712**	0.712	0.613
qdc	**0.637**	0.813	0.588	0.462	0.575	0.588	0.563	0.675	0.525	0.725	0.738	0.625
udc	**0.637**	0.813	0.588	0.462	0.575	0.588	0.563	0.675	0.525	0.725	0.738	0.625
klldc	0.738	0.725	**0.563**	**0.450**	0.625	**0.575**	**0.512**	0.650	0.600	0.750	0.762	**0.550**
pcldc	0.738	0.725	**0.563**	**0.450**	0.625	**0.575**	**0.512**	0.650	0.600	0.750	0.762	**0.550**
knnc	0.762	0.838	0.775	0.525	**0.000**	0.825	0.637	0.763	0.637	0.750	0.700	0.788
parzenc	0.688	0.725	0.613	0.512	0.575	0.600	0.575	0.688	0.512	0.738	**0.675**	0.650
treec	0.713	0.738	0.725	0.512	0.300	0.813	0.688	**0.600**	0.512	0.750	0.787	0.787
naivebc	0.688	0.738	0.662	0.488	0.400	0.587	0.588	0.675	**0.438**	0.838	0.725	0.700
svc	0.738	0.750	0.750	0.575	0.750	0.750	0.663	0.750	0.500	0.750	0.738	0.750
nusvc	0.675	**0.650**	0.600	0.512	0.600	0.663	0.587	0.675	0.550	0.738	0.762	0.650
mean	0.703	0.741	0.624	0.496	0.555	0.643	0.576	0.672	0.550	0.750	0.743	0.644

Principal Component Analysis

The percentages of explained cumulated variance and the classification performance for all numbers of principal components (scenario 2) have been given in Tables 6.13 and 6.14. The best results have been obtained for just 2 principal components, in terms of both CV and DPS mean errors (0.355 and 0.338 respectively). This is surprising as the first two components account for only 47.1% of the variance. Examination of PCA rotation matrix reveals that all original features are relevant as no weights are driven to zero.

Table 6.13: Classification errors for PCA–transformed dataset (10–fold CV)

classifier	Principal Component count										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
cum. variance	0.297	0.471	0.602	0.703	0.777	0.842	0.890	0.930	0.962	0.987	1.00
fisherc	0.500	0.390	0.378	0.343	0.354	0.371	0.392	0.386	0.345	0.360	0.350
ldc	**0.394**	0.339	0.403	0.334	0.341	0.303	**0.295**	0.354	**0.316**	0.300	**0.303**
loglc	0.460	0.318	0.426	0.365	0.346	0.335	0.341	**0.316**	0.327	0.376	0.420
nmc	**0.394**	0.314	0.361	0.355	0.371	0.359	0.365	0.388	0.338	0.343	0.336
nmsc	**0.394**	0.278	0.336	**0.313**	**0.336**	**0.300**	0.315	0.346	0.346	**0.289**	0.309
quadrc	0.404	0.290	0.398	0.418	0.424	0.444	0.514	0.617	0.560	0.525	0.524
qdc	0.433	0.313	0.409	0.414	0.416	0.424	0.471	0.546	0.533	0.591	0.540
udc	0.433	**0.273**	**0.314**	0.386	0.412	0.398	0.414	0.406	0.423	0.411	0.354
klldc	**0.394**	0.339	0.403	0.334	0.341	0.303	**0.295**	0.354	0.316	0.300	**0.303**
pcldc	**0.394**	0.339	0.403	0.334	0.341	0.303	**0.295**	0.354	0.316	0.300	**0.303**
knnc	0.475	0.340	0.370	0.361	0.409	0.455	0.415	0.468	0.485	0.455	0.451
parzenc	0.512	0.306	0.353	0.393	0.401	0.385	0.375	0.359	0.368	0.386	0.390
treec	0.464	0.526	0.557	0.560	0.555	0.564	0.580	0.571	0.573	0.593	0.597
naivebc	0.638	0.406	0.444	0.521	0.533	0.605	0.621	0.689	0.669	0.636	0.638
svc	0.523	0.485	0.455	0.418	0.420	0.418	0.380	0.388	0.391	0.365	0.360
nusvc	0.441	0.433	0.428	0.400	0.403	0.394	0.410	0.366	0.364	0.362	0.359
mean	0.453	0.355	0.402	0.390	0.400	0.397	0.405	0.432	0.417	0.412	0.408

Table 6.14: Classification errors for PCA–transformed dataset (8–fold DPS–S)

classifier	Principal Component count										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
cum. variance	0.297	0.471	0.602	0.703	0.777	0.842	0.890	0.930	0.962	0.987	1.00
fisherc	0.500	0.375	0.362	0.338	0.350	0.337	0.375	0.362	0.375	0.362	0.375
ldc	**0.388**	0.325	0.412	**0.250**	**0.250**	0.263	**0.237**	**0.250**	**0.300**	0.275	0.338
loglc	0.463	0.313	0.338	0.300	0.325	0.313	0.350	0.263	0.375	0.313	0.313
nmc	**0.388**	**0.250**	0.363	0.287	0.325	0.312	0.250	0.275	0.350	0.250	0.225
nmsc	**0.388**	0.275	0.363	**0.250**	0.300	**0.237**	0.250	0.263	0.313	**0.212**	**0.200**
quadrc	0.425	**0.250**	0.350	0.425	0.450	0.487	0.600	0.463	0.450	0.388	0.563
qdc	0.413	0.313	0.338	0.438	0.463	0.487	0.575	0.450	0.463	0.488	0.450
udc	0.413	0.300	**0.300**	0.325	0.313	0.350	0.413	0.350	0.388	0.350	0.287
klldc	**0.388**	0.325	0.412	**0.250**	**0.250**	0.263	**0.237**	**0.250**	**0.300**	0.275	0.338
pcldc	**0.388**	0.325	0.412	**0.250**	**0.250**	0.263	**0.237**	**0.250**	**0.300**	0.275	0.338
knnc	0.525	0.375	0.400	0.425	0.450	0.425	0.400	0.400	0.375	0.450	0.413
parzenc	0.563	0.287	0.350	0.388	0.400	0.362	0.362	0.375	0.375	0.375	0.375
treec	0.412	0.500	0.537	0.525	0.613	0.550	0.550	0.613	0.463	0.613	0.625
naivebc	0.575	0.375	0.438	0.587	0.587	0.537	0.588	0.588	0.613	0.650	0.637
svc	0.500	0.400	0.450	0.375	0.400	0.375	0.338	0.400	0.350	0.313	0.350
nusvc	0.475	0.425	0.350	0.362	0.362	0.287	0.325	0.262	0.338	0.225	**0.200**
mean	0.450	0.338	0.386	0.361	0.380	0.366	0.380	0.363	0.383	0.363	0.377

Linear Discriminant Analysis

An experiment similar to the one described in the previous section but using the Linear Discriminant Analysis has also been performed. Unlike PCA, LDA is a method which tries to find a linear projection of the data which best separates the classes and is thus useful for discrimination purposes. The shortcoming of LDA is that the maximum dimensionality of the projection is limited to $(c-1)$, where c is the number of classes [40].

Unfortunately, in the examined case there is no performance gain when compared to PCA and examination of the transformation matrix also does not indicate irrelevance of any of the original features, likely due to the high dimensionality reduction level.

6.3.5 Ensemble model

The final ensemble model has been designed using the generalised predictive system development cycle introduced in Section 2.2.2. As already discussed, the rationale behind using a combination of classifiers rather than a single best model is that various classifiers tend to differ for reasons ranging from different underlying mathematical models to different data or attributes used during the training process. This usually leads to a tendency of making classification errors on different instances. By taking advantage of this fact, it is possible to exploit this complementarity of various models and construct an ensemble able to outperform any individual classifier [98].

The following methods have been included in each of the pools depicted in Figure 2.4:

- **Base model pool** – 16 of the classifiers listed in Table B.1, excluding the Electrostatic Field Classifier for the reasons already discussed.
- **Preprocessor pool** – consisting of three groups of methods:
  - Three greedy feature selection methods: forward, backward and plus–L–takeaway–R feature selection, all executed within a 10–fold CV scheme without repeating, allowing for randomness leading to greater diversity. For this very reason DPS was not used at this stage due to its deterministic nature. Error rate of each base classifier has been in turn used as a criterion for feature selection, the procedure has been repeated 10 times and candidate classifiers have been created using all obtained unique feature subset/base classifier pairs.

    Feature selection has been chosen over transformation (i.e. PCA, LDA) for a number of reasons. First of all, the latter approach did not demonstrate a considerable performance improvement at the same time bringing in the loss of interpretability of the results. Moreover, feature selection may facilitate reduction of data acquisition costs – if some attributes are never used, there is no need to measure them by performing expensive biological tests.
  - Maximum likelihood imputation from univariate class conditional distributions rather than mean imputation. The procedure involved estimation of the probability density function for each class/feature pair using the Parzen window method [40] and imputation of the most likely value from this distribution. Figure 6.5 depicts an example of how the value imputed using this approach may vary from the class conditional mean. In further experiments this imputation method allowed to achieve on average 0.025 10–fold CV error improvement using all 11 features, with the most improved classifier (*loglc*) better by 0.074.
  - Missingness model, which creates a binary missingness map for the training data, denoting a missing value by 1. The columns with all 0's are then dropped (they correspond to the features of the original dataset which are never missing) and the training dataset is augmented with the remaining part of the missingness map by treating each column of the map as a new feature. An average 10–fold CV error improvement of 0.050 due to using the described missingness model was observed, with the most improved classifier (*quadr*) better by as much as 0.288.

  For more details on the preprocessing techniques used please refer to Appendix B.3.
- **Error estimator pool** – consisting of the following four methods:
  - 10–fold cross–validation.

  - Nested 10–fold cross–validation (NCV) presented in Figure 6.6. In the 10–fold nested cross–validation scheme the whole dataset is first randomly divided into 10 test folds, in this case consisting of 5 objects each (1 object per class). Then each of those 10 test folds is in turn put aside as test data and a number of combined models is constructed using the remaining 9 folds (now called validation folds) in a similar, iterative manner: each of the 9 validation folds is in turn put aside, all candidate classifiers are trained on the remaining 8 folds and tested on the validation fold. After iterating over all 9 validation folds, a binary validation error map is constructed for each of the candidate classifiers and the validation set errors are calculated. The procedure is repeated for all 10 test folds and the result are two error maps: (1) validation, which can be used for selecting candidate classifiers for inclusion in an ensemble, and (2) test, which can be used to calculate the out–of–sample error of the final ensemble.
  - 8–fold supervised Density Preserving Sampling (DPS–S).
  - Nested 8–fold Density Preserving Sampling (NDPS), which is analogous to NCV, except it is not repeated and the number of folds has been set to 8.

  In the original study published in [21] NCV had been used, as the DPS method had not yet been developed. In Section 6.3.6 however, the experimental results based on NDPS have also been included for comparison.

- **Postprocessor pool** – consisting of a single Majority Voting method, which is simple and fast, as it operates on binary correct/incorrect values and has been thoroughly researched in the past (see [138]).
- **Model selection criterion pool** – single criterion, selecting 21 top performing candidate models in terms of the nested CV/DPS error for exhaustive search for the best combination, and 20 top performing ensembles for level 2 combinations, if a multistage structure is constructed. The computational and space requirements of the exhaustive search are the factors limiting the number of considered candidate models.

6.3.6 Experimental results

The generalisation performance of built models estimated by the classification errors on the test set have been given in Tables 6.15 (NCV) and 6.16 (NDPS). For NCV this is equivalent to 10 times repeated 10–fold cross–validation error and 8–fold DPS error in the case of NDPS. Since the two errors cannot be compared directly, another column has been added to Table 6.16, containing 10–fold CV error of the obtained ensembles.

For the original experiment using the nested cross–validation scheme there were 569 candidate classifiers in total, with mean error of 0.311. The test error of the best component classifier (*parzenc* built on features 1, 3, 4, 7, 9 and 12) was 0.180, which compared to the previous results from Table 6.6 (0.29, all features used) and Table 6.13 (0.273, PCA) is a considerable improvement. The same classifier has produced the lowest validation error of 0.203 and although this does not necessarily imply the lowest test error, the two types of errors are well correlated, as will be shown later. Note, that at this stage the MV error (0.218) did not improve over the error of the best classifier.

As mentioned earlier, due to limited resources only 21 best candidate classifiers in terms of the validation error have been selected for combining. Their MV test error is equal to 0.164, which for the first time is less than the error of the best candidate classifier. Exhaustive search for the best

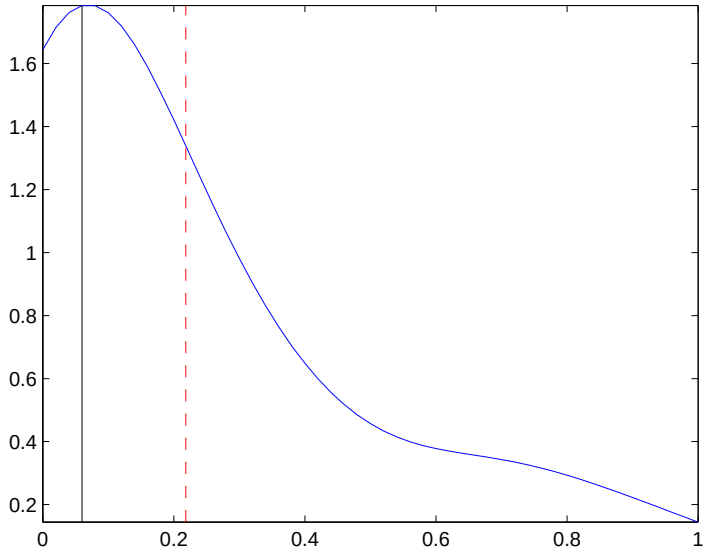


Figure 6.5: Mean imputation (dashed vertical line) v. imputation from univariate distribution (solid vertical line) for the 10^{th} feature, superimposed on the PDF (blue solid line)

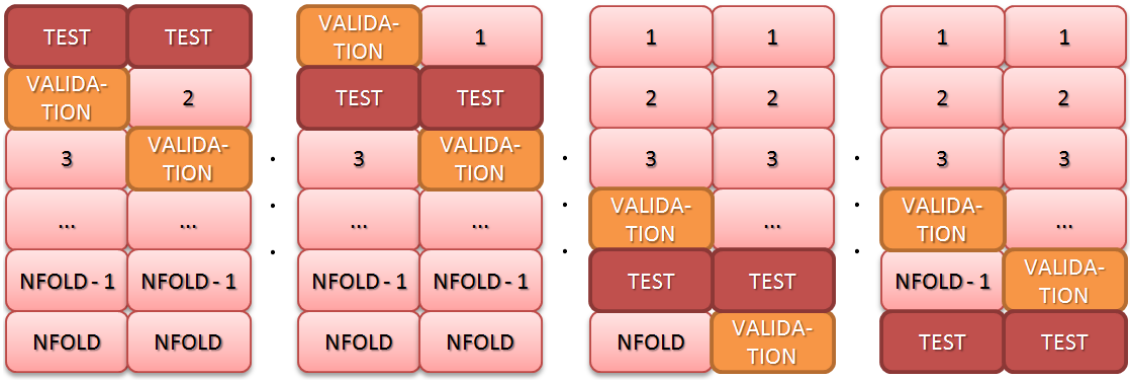


Figure 6.6: Nested cross–validation scheme. NFOLD denotes the total number of folds.

combination of 21 candidate classifiers (level 1 combinations) brings further improvements. Test error of the best level 1 combination is 0.132. This time however the best test set model is only 6286^{th} in the validation data performance ranking of combinations and thus there is no reason to prefer this particular model over the rest. The test error of the highest ranked combination is 0.132 and the MV error of a subset of best level 1 combinations (with validation error within one standard deviation from the minimal validation error) is 0.144.

For the level 2 combinations, 20 best level 1 models have been chosen and once again combined exhaustively. This time the best combination produced 0.102 test error (0.136 validation error, 96^{th} in the ranking) and the lowest MV error achieved was 0.106. An attempt for another combination level did not produce further improvements.

In the case of the nested DPS scheme (Table 6.16), for the reasons already discussed, the same 569 classifiers have been used as a starting point. The best component and candidate classifier in

Table 6.15: Test errors of combinations, component and candidate classifiers (NCV)

test error	min	mean	max	MV[a]	count
candidate classifiers	0.180	0.311	0.573	**0.218**	569
component classifiers	0.180	0.223	0.264	**0.164**	21
level 1 combinations (all)	0.132	0.169	0.226	**0.166**	1 048 576
level 1 combinations (better than mean)	0.132	0.165	0.192	**0.164**	558 072
level 1 combinations (better than min+stdev)	0.136	0.154	0.172	**0.144**	1 381
level 2 combinations (all)	0.102	0.113	0.128	**0.120**	524 288
level 2 combinations (better than mean)	0.102	0.112	0.126	**0.110**	215 222
level 2 combinations (better than min+stdev)	0.104	0.109	0.116	**0.106**	95

[a] 10 times repeated 10–fold CV Majority Voting error

Table 6.16: Test errors of combinations, component and candidate classifiers (NDPS)

test error	min	mean	max	MV[a]	MV[b]	count
candidate classifiers	0.160	0.307	0.600	**0.200**	**0.218**	569
component classifiers	0.160	0.198	0.240	**0.160**	**0.164**	21
level 1 combinations (all)	0.100	0.160	0.240	**0.140**	**0.120**	1 048 576
level 1 combinations (better than mean)	0.100	0.154	0.240	**0.140**	**0.120**	645 505
level 1 combinations (better than min+stdev)	0.100	0.143	0.220	**0.120**	**0.115**	32 557

[a] 8–fold DPS Majority Voting error
[b] 10 times repeated 10–fold CV Majority Voting error

terms of the 8–fold DPS error (*ldc* using features 2, 3, 4, 7, 9, 11 and 12) is however not the same as in the case of NCV. Moreover, when it comes to the ensemble models, the situation looks even more differently, as only level 1 combinations have been built since additional levels did not bring any improvements. This was however sufficient to obtain a combination which performs closely to the one built using NCV (0.115 v. 0.106) in terms of the 10–fold CV error. Note, that this has been achieved with considerable computational savings when compared to the NCV approach, as the nested DPS procedure did not need to be repeated 10 times and there was no need to build level 2 ensembles.

Correlation between test and validation error

During selection of models to be combined, the validation error has been used as the criterion. To confirm that it was indeed the right choice, correlation between the two types of errors for the NCV approach was calculated. At the level of individual candidate classifiers the correlation is very high (0.9662), but it drops considerably for level 1 combinations (0.6076) just to decrease even further for level 2 combinations (0.4889). This can also be seen on the plots of test v. validation errors given in Figure 6.7. First thing to notice is that by combining individual models both errors have been dragged towards the origin and are much more concentrated – the variability of error in the pool of models is smaller. Also, the test error is on average smaller than the validation error which is due to the amount of data used for training (45 objects in the case of test set and 40 in the case of the validation set). Note that high error correlation of individual

classifiers is partly caused by much wider range of validation errors than for combinations on both levels, while the ranges of test errors are similar.

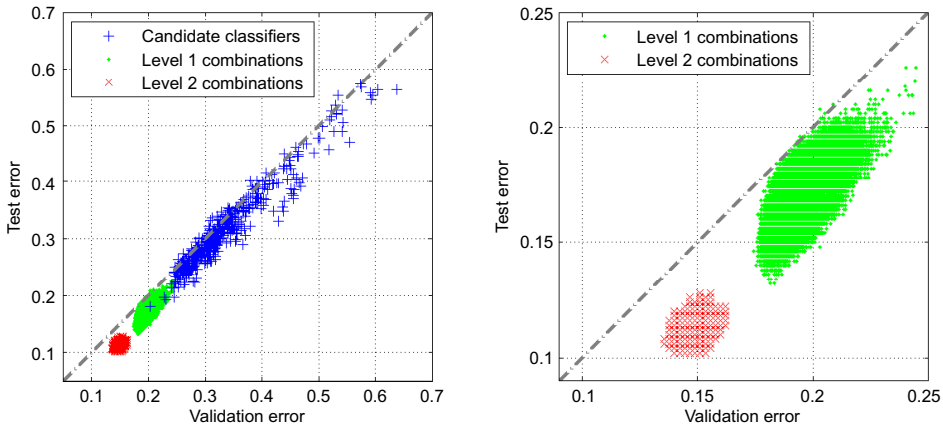


Figure 6.7: Test versus validation errors (mean values of repeated cross–validation)

Figure 6.8 depicts test v. validation errors again, this time represented by ellipses with semiaxes equal to standard deviations of respective errors. It can be well seen especially on the close–up plot, that apart from reducing the mean error value, the variance of test error has also diminished. The plots for the NDPS approach are not presented here as they look very similar.

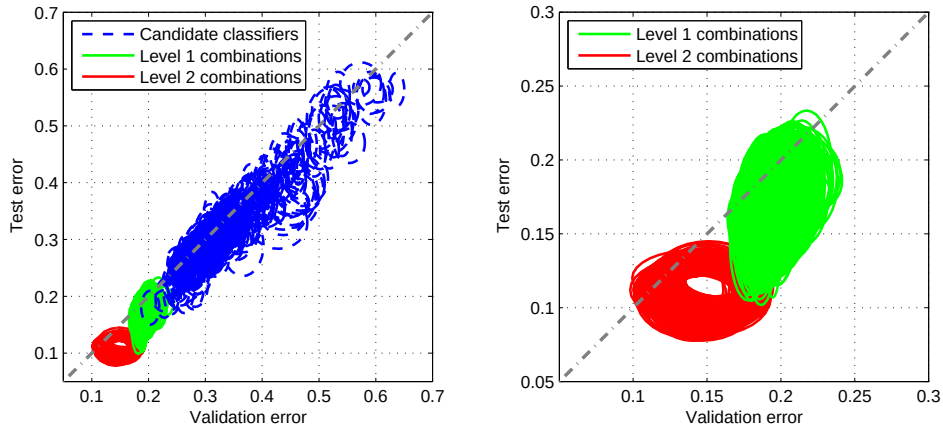


Figure 6.8: Test versus validation errors (mean values + stdev of repeated cross–validation)

Difficult objects

To understand where do the errors come from a plot of misclassification rates of the 95 best level 2 NCV combinations has been given in Figure 6.9. Each bar represents the percentage of level 2 models which have misclassified a particular object (objects 1 to 20 represent the control site, 21 to 30 class T4-S1 etc.). Due to the combination method used, any object with misclassification rate ≥ 0.5 will be misclassified by the ensemble as well. As it can be seen, there are 5 such objects in the dataset which corresponds to 0.100 error (the actual error is 0.106 as it has been averaged over 10 runs and so were the misclassification rates).

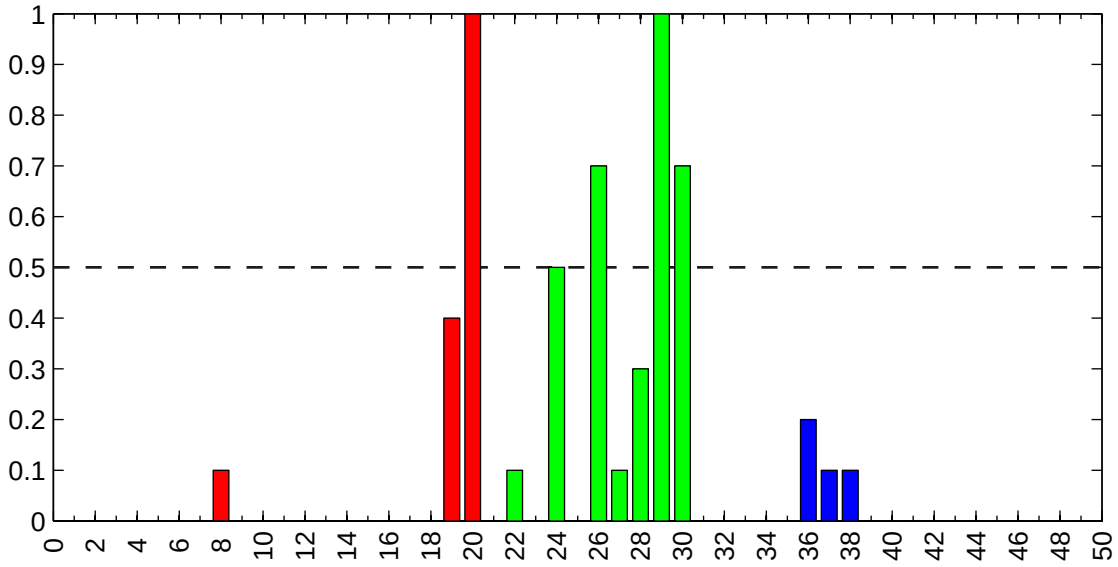


Figure 6.9: Misclassification rates of the best level 2 models for each test object (NCV)

Classification of objects belonging to the heavily polluted site (objects 41 through 50) does not pose a problem, as only single models make errors there. The same applies to most of the instances belonging to the moderately polluted class as only for 3 of them the models slightly disagree regarding the class label. Also most of the instances belonging to the control site are classified correctly without difficulty. There are two exceptions however – instance 20 is never classified correctly and instance 19 is classified correctly only by a small margin. The class causing most problems is the lightly polluted site with 4 (i.e. 40%) of instances being misclassified, 3 of which by a considerable margin. This allows to presume, that although the biomarkers used in this study facilitate good discrimination between the classes in most cases, they are not discriminative enough to separate the objects coming from the lightly polluted site (T4-S1) and the control site (T0-C + T4-C). The pollution level at site T4-S1 might be so low, that these particular biomarkers fail to detect it. Thus a possible future research direction is to focus on the evaluation of usability of various biomarkers for detection of very low pollution levels.

Feature usage

The usage of particular features by all component classifiers included in the top 95 level 2 NCV combinations has been depicted in Figure 6.10. As it can be seen, apart from feature 5, which has been dropped deliberately, there is another attribute which appears irrelevant – 10. Also the attribute number 1 is used by less than 20% of component classifiers, so it might be considered as the second candidate for removal. The rest of the features appears important, with attribute 9 absolutely crucial, as it has been used by all component classifiers.

The above confirms that the biomarkers selected to form the input of the predictive system were suitable for the task, as there is no doubt that 9 out of 12 attributes are relevant. Although the choice was good, it was not necessarily optimal – it is possible that some other set of biomarkers would even facilitate error–free classification. Unfortunately, it is almost impossible to tell which biomarkers should be used for a particular task before the actual model is built. Thus in a perfect world scenario one would run as many biological tests as possible and then select an optimal subset of attributes during the model building process. There are two limiting factors

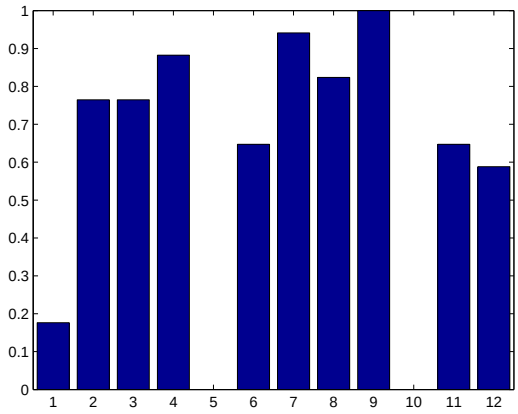


Figure 6.10: Feature usage by component classifiers

here however. Each biological test has an associated monetary cost and a total cost of performing a batch of tests is limited. Moreover, due to ethical reasons the amount of biological material used should be as small as possible, to leave the environment in its original state. This leads to an interesting problem of balancing the classification performance with the total cost (both monetary and ethical) of performing a set of biological tests. In other words, the ability to estimate how much would it cost to achieve a given performance level or how accurate can the predictive model be given a known cost limit, might be very desirable. Such analysis could even reveal that the best performance might be achieved using only a small set of relatively inexpensive tests, which would minimise both the total cost and classification error at the same time. Although with the limited amount of data it was not possible to address this problem here, it is a promising research direction, worth considering in future studies.

6.3.7 Summary of case study findings

Coastal water pollution monitoring using the biomarker data is very appealing as the biomarkers are able to detect even a very low concentration of pollutants, unobservable using different methods. Due to the specific data collection process however, the biomarker data is rather difficult to process in order to obtain meaningful results. Blindly applying one of the many available machine learning techniques is seldom successful, thus a principled approach in a form of the predictive system development cycle is crucial. The most important conclusions are:

- biomarkers can be successfully used for discrimination between various aquatic toxicity levels even when small amount of data is available,
- a sophisticated multi–stage ensemble model had to be built, to deal with imperfections of the data (limited amount and low quality),
- the choice of biomarkers for a specific predictive task can dramatically influence performance of the built models and is also strongly connected with the monetary cost and ethical issues of data acquisition,
- due to natural evolution of the environment regardless of the changes in pollution level, the results could be further improved if some environmental features were also measured (e.g. water temperature).

Although the predictive model obtained by following the generalised predictive system development cycle derived in Section 2.2.2 performs relatively well, some open problems remain. First, it is still not known how to select the biological tests to be performed, before building the prototype of the model (i.e. before actually performing the tests). Since the literature on using biomarkers for predictive modelling is discordant, it seems that at least for some time this choice will need to remain more or less random. This issue leads to another interesting problem of balancing the model performance with data acquisition cost. It might be the case that for some applications, models cheap to develop yet not very accurate would be sufficient, while for some other purpose the precision of the model will be the most important factor, regardless of the cost.

6.4 Concluding remarks

In this chapter the techniques developed within the thesis have been confronted with two real–world environmental problems.

The generalised predictive system development cycle described in Section 2.2.2 has proven to be very successful. It's application within the Environmental Toxicity Prediction Challenge CADASTER 2009 resulted in a predictive model awarded a title of the First–Pass Winner, outperforming over 100 other submissions from all over the world. The second application of the proposed design cycle to predictive modelling of water pollution using low quality biomarker data can also be considered a success, as some of the outcomes were publication of the results in a respected international journal[15] [21] and a joint conference paper [119].

The performance of Electrostatic Field Classifier on the biomarker data is disappointing, although it performed very well in the experiments using a range of benchmark datasets (Section 3.6). A possible explanation can be sought by looking at the behaviour of the decision tree classifier (*treec*), which is the worst performing base model in the water pollution prediction problem, despite being considered as state–of–the–art in some areas, where interpretability of the results is essential. It stems from one of the interpretations of the 'No free lunch theorem' [40], which states that if a base model is exceptionally good in some applications, it must be exceptionally bad in some other applications. In other words, there are no universally excellent models. It thus seems that the problem of water pollution modelling using biomarker data lies somewhere in the space of all possible problems, where both EFC and decision trees are not that strong.

The Density Preserving Sampling technique has demonstrated its potential in both environmental applications discussed in this chapter, leading to development of models performing at the level comparable to standard cross–validation but requiring a fraction of computations of the latter. Should thus DPS completely replace CV?

One needs to have in mind that in its current form DPS is based on greedy and hence suboptimal optimisation. Although in all the experiments included in this thesis the technique performed well, there is no guarantee that this will always be the case. For this reason, until DPS has been thoroughly verified in a number of practical applications (inclusion of the technique into PRTools is an important step in this direction), it should be used with caution. Cross–validation on the other hand is a method, which has been around for more than 40 years and has been successfully applied to countless problems, not only in machine learning but also statistics. Hence at this stage, DPS should be perceived as a solution complementary to CV rather than its replacement, especially in safety–critical applications.

[15] Elsevier's Water Research with 5–year impact factor of 4.828

Chapter 7

Conclusions

7.1 Project summary

The primary objective of this research was to investigate and explore some of the similarities between physical world and computational intelligence in order to find inspirations and design new nature inspired machine learning techniques.

The literature review undertaken at the beginning of this research project quickly revealed that the number of physically inspired machine learning methods of practical importance is relatively low. Moreover, many of these techniques have evolved into a suite of specific, constrained machine learning models, each having its strengths and weaknesses, which in separation could only be applied to a limited number of problems. The Information Theoretic Learning framework introduced in Chapter 2 is the only identified attempt of an algorithm independent (and thus having a wider range of possible applications) development based on physical analogy. However, at this stage the lack of a high–level, algorithm independent learning technique, spanning the whole suite of machine learning methods became apparent.

At the highest level this issue has been addressed in Chapter 2, in a form of the generalised predictive system development cycle, being in fact a methodology for development of well–performing data–driven predictive models. Although the proposed cycle is not inspired by any physical phenomena, it forms an essential algorithm independent learning framework, enabling comprehensive validation and evaluation of techniques developed in later chapters of this thesis. Also, as demonstrated in Chapter 6, the proposed methodology is of considerable practical importance, allowing to design predictive systems without having specific domain knowledge, yet also without sacrificing their performance.

The Electrostatic Field Classification Framework for supervised and semi–supervised learning from incomplete data described in Chapter 3 is the first physically inspired machine learning method proposed in this thesis. The Framework stems from a direct analogy to potential fields, where each data instance is treated as a charged particle. The classification is performed in a simulatory manner by allowing the 'data particles' to interact according to a set of simple, clearly defined rules, until convergence. However, instead of designing yet another classification algorithm, the focus has been put on a much more general problem – learning from incomplete data. The Electrostatic Field Classification Framework not only supports classification of data instances with missing attributes, but also learning from deficient datasets, including the missing class label scenario. The novelty in this approach lies in extension of the electrostatic field analogy to naturally handle various deficient data situations within a unified classification framework, which in addition is extendible, allowing for example to incorporate a categorical attribute

handling routine.

A different research direction, this time exploiting indirect physical analogy in a form of the ITL framework has been pursued in Chapter 4. The resultant novel Density Preserving Sampling technique is another algorithm independent learning development, with a wide area of applications traditionally dominated by the cross–validation. DPS samples the dataset in order to maximise the correntropy, a recently introduced ITL measure of the probability density function similarity. This results in splits of the data, which are maximally representative in the correntropy sense as well as deterministic, thus there is no need to repeat the sampling procedure multiple times. The DPS splits can be used for model generalisation error estimation, producing results very similar to cross–validation at a fraction of computations required by repeated CV and alleviating the risk connected with running the cross–validation procedure only once. As demonstrated in the experiments, DPS is also well suited for model ranking and selection, and there are other potential applications, which however require further investigation.

DPS has brought a reduction in required computational resources by an order of magnitude when compared to 10–times repeated cross–validation but as discussed in Chapter 4, further reductions are possible. Hence inspired by the success of the DPS technique, in Chapter 5 the usability of various PDF divergence measures, including the ITL–based Cauchy–Schwarz divergence, for the purpose of representative sampling has been investigated. The goal has been set to limit the computational requirements of the generalisation error estimation procedure by another order of magnitude. The experiments have confirmed that it is indeed possible but none of the examined divergence measures is suitable for this purpose. On this occasion a number of divergence estimation methods have also been investigated, questioning their usability for datasets consisting of less than thousands of instances.

The physically inspired techniques developed in this thesis perfectly fit within the proposed generalised predictive system development cycle. For that reason they have all been evaluated together in Chapter 6, using two environmental case studies. The cycle itself has proven to be a success, as a purely data–driven toxicity prediction system developed with its help has been awarded a title of the First–Pass Winner in the Environmental Toxicity Prediction Challenge CADASTER 2009. The proposed methodology has also been successfully used in order to develop a multistage ensemble model for water pollution prediction using low–quality biomarker data. Although at the time of development of the the two systems mentioned above the DPS technique had not been derived yet, for comparison purposes the experimental results of Chapter 6 have been supplemented to include DPS, revealing exceptional performance in both cases. More specifically, the systems developed using DPS instead of CV have performed at a very similar level, saving almost 90% of computations at the same time.

As it can be seen, physical models have proven to be a fruitful source of inspirations for computational intelligence, thus the main objective of this research project has been achieved. The techniques developed in this thesis have a wide range of possible applications, but one should also be aware of their limitations. For the machine learning techniques being a direct analogy to physical phenomena (i.e. EFCF), the dimensionality of the input space appears to be the biggest obstacle. This shouldn't be very surprising as the real physical fields act in the real 3–dimensional physical world. In high–dimensional sparse spaces, consisting mainly of void, the interactions are necessarily very weak, which coupled with a limited precision of computations makes the extension of the models to higher dimensions not that straightforward. Fortunately it is not impossible either, but the required modification of the distance definition is in fact a departure from the real physical model, weakening the physical analogy at the same time and naturally leading to more indirect inspirations, like the ITL.

Indeed the ITL techniques seem to behave much better as the dimensionality of the input space increases, although it should still be kept within a reason. This however shouldn't be perceived as a limitation of ITL per se, as the 'curse of dimensionality' is a well known phenomenon concerning machine learning and other disciplines in general. Nevertheless ITL does have its own weaknesses, resulting from its strong dependance on the PDFs, rendering ITL useless without good estimators of the latter. And PDF estimation is an important research area on its own.

The above shows, that although physically inspired learning definitely is a subject area worth researching, it comes with its own set of issues and difficulties, and is hence not able to solve all problems of the machine learning world on its own. But is any other technique or suite of techniques able to do so? The importance of physically inspired models thus stems from the fact that they are different and do not aim at replacing the existing methods but rather to complement them. After all, as discussed multiple times, diversity is one of the key factors of success in predictive modelling.

7.2 Main findings and contributions

The original contributions of this work are:

- **Rigorous generalised predictive system development cycle.**

  The cycle has been developed in order to address the weaknesses of the classical predictive system development cycle which can be found in many machine learning textbooks. The main characteristic of the proposed cycle is that it allows to build well–performing predictive models, even in the absence of domain knowledge, thus in a purely data–driven setting. Moreover, the obtained models can successfully compete with the ones developed by the experts in a given domain, which is achieved by offsetting the lack of the problem domain knowledge by additional computations and automation of activities which are usually performed manually. The proposed methodology has been verified on two real–world environmental problems. One of them was the Environmental Toxicity Prediction Challenge CADASTER 2009, in which the predictive system developed in a purely data–driven way has been awarded as the First–Pass winner, non–significantly different from the best performing submission developed by an expert in QSAR modelling.

- **Comprehensive Electrostatic Field Classification Framework for supervised and semi–supervised learning from incomplete data.**

  The framework has been derived by taking advantage of a strong, direct analogy to physical potential fields, building upon two existing classification techniques: the Gravity Field Classifier and the Electrostatic Field Classifier. The framework tackles the problem of learning from incomplete data in a comprehensive way, as the incompleteness is understood in a very broad sense including (1) missing attribute values in the test data, (2) missing attribute values in the training data, (3) missing labels in the training data, or (4) any combination of the above. One of the important features of the proposed framework is its modular, extendible architecture, allowing to easily incorporate alternative definitions of force and distance.

  In this thesis some previously not addressed issues of the two classifiers have also been investigated, with the distance concentration phenomenon being the most important one. As it turned out, the models tend to perform very badly in higher dimensional spaces. This most probably stems from their origin in a form of physical fields, which exists only in three

dimensions. The issue has been addressed by departing from a strict physical definition of the field towards a more loose analogy, with a great success.

The Electrostatic Field Classification Framework fits within the base classifier and preprocessor level of the proposed generalised predictive system design cycle.

- **The Density Preserving Sampling technique.**

  The algorithm independent learning technique, which has been derived as an alternative to commonly used cross–validation by taking advantage of an indirect analogy to physical phenomena in the form of the Information Theoretic Learning framework. DPS is a data splitting method, which aims at producing splits or folds which are representative. This is facilitated by a sampling procedure guided by optimisation of the correntropy, a recently introduced ITL probability density function similarity measure. DPS allows to reduce the computational requirements of generalisation error estimation procedure by an order of magnitude when compared to cross–validation, without compromising quality of the estimate and protecting against typical pitfalls connected with random sampling. The technique has also proven very useful for model ranking and selection, which has been confirmed by applying it to two real–world environmental problems mentioned before.

  The Density Preserving Sampling technique fits within the error estimator level of the proposed generalised predictive system design cycle.

- **Experimental comparative study of PDF divergence estimators and their usability in sampling for generalisation error estimation, including an attempt to gather relevant concepts in the area of divergence estimation and present them in a unified way.**

  Encouraged by the good performance of DPS, the goal of this study was to investigate if it is possible to go one step further and estimate model generalisation ability by testing it only once, using a single, carefully selected subset of data. The first part of the study, devoted to evaluation of PDF divergence estimators revealed, that in many cases datasets consisting of thousands of instances are needed for accurate divergence estimation. This has put the usability of divergence estimators in question, especially after examining some of their applications which can be found in the literature.

  In the second part of the study 70 divergence estimators have been used in order to assess correlation between their values and the bias of the obtained generalisation error estimate. Although the correlation has been observed in a limited number of cases, in general the dependency is not strong and frequent enough for this approach to have a practical meaning.

- **Multi–stage Multiple Classifier System for robust predictive modelling of water pollution using biomarker data.**

  A real–world application, in which all techniques proposed in this thesis come together. Despite confirming the usability of the generalised predictive system development cycle and the DPS technique and incorporation of a Missing Not at Random data modelling approach, this study tackled an important problem of learning from low–quality biomarker data. The process of data acquisition in environmental sciences, due to many physical and ethical constraints is very specific and often results in datasets, which are difficult to process by data analysts in order to obtain meaningful results. Many such problems have been identified in this study and publication of the results in a respected international journal devoted to environmental sciences allows to hope, that the data acquisition procedures will change in the future, wherever possible.

7.3 Further research

Further research directions will revolve around two themes which have been investigated in this thesis. The first theme is the ITL framework. This theme is more of a theoretical nature as the research efforts will not be directed at any particular real–world application. Within this theme the following topics will be addressed:

- **The Density Preserving Sampling technique.**

  As mentioned before, greedy optimisation of correntropy within the DPS sampling procedure is currently perceived as one of its weakest points. Hence employment of other optimisation techniques might help to alleviate the risk of being caught in local minima of the objective function. An important issue is however for the optimisation process to be fast and efficient, so that it does not cancel out the computational savings DPS brings over cross–validation in the first place. For that reason the Genetic Algorithms cannot be used here and the only feasible procedure is the Lagrange optimisation facilitated by introducing soft memberships of instances in the obtained subsamples, in a way similar to [83].

  Another limitation of current DPS is that at each stage only two equal splits are produced. If more than two splits are required, which is usually the case, the procedure needs to be repeated. Although this does not increase the computational complexity considerably, as the pairwise distances only need to be calculated once, the total number of splits obtained must be a power of 2. Hence development of a multi–split version of DPS is worthwhile.

  Finally, investigation of the behaviour of DPS in conjunction with early stopping for training of e.g. Artificial Neural Networks supplements the list of possible applications examined in this thesis.

- **Application of ITL estimators of mutual information to sampling and other problems.**

  Two easy to calculate estimators of mutual information have been proposed in [127]: Cauchy–Schwarz quadratic mutual information and Euclidean distance quadratic mutual information. Their details can be found in Appendix C. Since MI alongside entropy is one of the most important concepts of information theory, the number of possible applications is almost unlimited. Surprisingly, although the two estimators have been around for 10 years now, only a few published applications exist, generally focusing on either Blind Source Separation [73, 74] or feature extraction in classification problems [166, 167, 168, 169, 170, 165]). Since the attempt to use PDF divergence measures to guide sampling described in Chapter 5 was not entirely successful, another natural choice to try is mutual information as an objective function to optimise. Exploration of other possible applications, like feature extraction for regression or unsupervised dimensionality reduction are also promising research directions.

- **Application of ITL to building ensemble models.**

  One of the areas of the generalised predictive system design cycle proposed in Chapter 2 which has not been targeted in this thesis is the postprocessor pool. With respect to classification problems, this issue will be addressed by development of a combiner, which maximises the classification margin. According to [152] margin maximisation allows to reduce the generalisation error by adding additional ensemble members long after the training error has vanished. However, in order to achieve this, apart from maximising the margin it is important for the margin PDF to assume appropriate shape. The ITL

framework with its entropy, correntropy and mutual information manipulation capabilities is well suited for this goal. Unlike other approaches which focus on manipulating the weights of ensemble members in the final decision, a combiner driven by an ITL criterion would actively influence the whole process of ensemble members training in order to obtain the desired margin PDF shape.

The second, more practical future research theme stems from successful collaboration with the International Research Institute of Stavanger (IRIS) in Norway, which has resulted in the water pollution monitoring study described in Chapter 6. As mentioned before, the study has left numerous questions and issues, which need to be addressed:

- **Modification of the biomarker data acquisition procedure** to eliminate the most obvious causes of inconsistent modelling outcomes (e.g. instance pairing) and to arrive at a procedure acceptable for both biochemists and data analysts.
- **Addressing the concept drift issue** by incorporating appropriate algorithms into the current predictive system and including various environmental parameters into the set of input variables, as this would allow to detect changes in the data which are not due to changes in the pollution level.
- **Investigation of the cost–performance trade–off,** connected with varying monetary cost of different biochemical tests. As already discussed, performing all possible tests on each organism is not feasible not only due to the monetary cost but also because many tests are destructive or mutually exclusive. On the other hand the more complete the set of attributes to choose from when building a predictive system, the better results can be expected. A possible solution might be found by using meta–learning techniques in order to exploit successes or failures of other researchers working in the area of environmental monitoring.

The final future research topic is connected with the Electrostatic Field Classification Framework described in Chapter 3, and more specifically with the distance concentration phenomenon so apparent in the operation of the two classifiers used. In the experiments performed the order of the distance measure used has been derived by exhaustive search, which obviously is not efficient. Hence some more principled mechanism for determination of this parameter is needed. Once again meta–learning might be a possible solution, but this issue still remains to be investigated.

Appendix A

Benchmark datasets

Throughout this thesis a number of datasets has been used in order to illustrate various concepts as well as to assess the performance of proposed methods and models in a comprehensive way. In order to make the results comparable with those available in the current and future literature, many experiments have been performed using publicly available benchmark datasets. The datasets come from the UCI Machine Learning Repository [6], the UCL Machine Learning Group's ELENA database[1], the PRTools Pattern Recognition Toolbox for MATLAB [42] as well as two standard textbooks [97, 134]. The datasets were selected according to their size, dimensionality and the number of classes to form a diverse blend, enabling a comprehensive evaluation of discussed techniques. A summary of the benchmark datasets used have been given in Table A.1 and a detailed description follows.

1. **AZI**
   Azizah dataset, which is concerned with segment features in utility symbols and is available as a part of PRTools.

2. **BIO**
   Biomed dataset, which is concerned with biomedical diagnostics and is available as a part of PRTools.

3. **CAN**
   The breast cancer database, which is available at the UCI repository and has been obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. This dataset was used to distinguish between benign and malignant breast cancers, based on features computed from a digitised image of a fine needle aspirate of a breast mass.

4. **CBA**
   Chromosome banding patterns data, which is available as a part of PRTools. The original dataset consists of 12000 instances, 1000 of which have been selected for experiments.

5. **CHR**
   Chromosomes recognition dataset, which is available as a part of PRTools.

6. **CLO**
   The clouds dataset, which is an artificial dataset available at the ELENA database, dedicated to the study of classifier behaviour for heavy intersection of the class distributions and for

[1] http://www.dice.ucl.ac.be/mlg/?page=Elena

Table A.1: Benchmark datasets summary

no.	acronym	name	source	# objects	# attributes	# classes
1	azi	Azizah dataset	PRTools	291	8	20
2	bio	Biomedical diagnosis	PRTools	194	5	2
3	can	Breast cancer Wisconsin	UCI	569	30	2
4	cba	Chromosome bands	PRTools	1000*	30	24
5	chr	Chromosome	PRTools	1143	8	24
6	clo	Clouds	ELENA	1000*	2	2
7	cnc	Concentric	ELENA	1000*	2	2
8	cnt	Cone–torus	[97]	800	3	2
9	dia	Pima Indians diabetes	UCI	768	8	2
10	ga2	Gaussians 2d	ELENA	1000*	2	2
11	ga4	Gaussians 4d	ELENA	1000*	4	2
12	ga8	Gaussians 8d	ELENA	1000*	8	2
13	gla	Glass identification data	UCI	214	10	6
14	ion	Ionosphere radar data	UCI	351	34	2
15	iri	Iris dataset	UCI	150	4	3
16	let	Letter images	UCI	1000*	16	26
17	liv	Liver disorder	UCI	345	6	2
18	pho	Phoneme speech	ELENA	1000*	5	2
19	sat	Satellite images	UCI	1000*	36	6
20	seg	Image segmentation	UCI	1000*	19	7
21	shu	Shuttle	UCI	1000*	9	7
22	son	Sonar signal database	UCI	208	60	2
23	syn	Synth–mat	[134]	1250	2	2
24	tex	Texture	ELENA	1000*	40	11
25	thy	Thyroid gland data	UCI	215	5	3
26	veh	Vehicle silhouettes	UCI	846	18	4
27	win	Wine recognition data	UCI	178	13	3

* The number of instances actually used in the experiments, selected randomly with stratified sampling from the whole, much larger dataset in order to keep the experiments computationally tractable.

high degree of nonlinearity of the class boundaries. The data represents 2–dimensional distributions with two classes. Class 0 was obtained from the sum of three different Gaussian distributions, while class 1 represents a single normal distribution. The original dataset consists of 5000 instances, 1000 of which have been selected for experiments.

7. **CNC**
   The concentric dataset, which is an artificial dataset available at the ELENA database representing the study of linear separability of a classifier when some classes are nested in other without overlapping. The 2–dimensional dataset represents uniform concentric circular distributions with two classes. The dataset is entirely contained in the square (0,0), (1,1). The points of class 0 are uniformly distributed into a circle of radius 0.3 centred on (0.5,0.5). The points of class 1 are uniformly distributed into a ring centered on (0.5,0.5) with internal and external radius respectively equal to 0.3 and 0.5. The original dataset consists of 2500 instances, 1000 of which have been selected for experiments.

8. **CNT**
   Cone–torus, which is a synthetic 2–dimensional dataset first used in [97] and consisting of 3 classes, with data points generated from 3 differently shaped distributions: a cone, half a torus, and a normal distribution with prior probabilities of 0.25, 0.25 and 0.5 respectively. The dataset comes divided into two non–overlapping parts (training and test) with 400 instances each.

9. **DIA**
   Pima Indians diabetes database, which is available at the UCI repository, providing diabetic diagnosis based on a number of various physiological attributes of the examined patients.

10. **GA2, GA4, GA8**
    The Gaussian database, which is a group of artificial datasets available at the ELENA database, concerned with the study of classifier behaviour for different dimensionalities of the input vector, under heavily overlapped distributions with no linear separability. All datasets represent the same 2–class problems but with dimensionality ranging from 2 to 8. Class 0 represents multivariate normal distribution with zero mean and unit standard deviation, while class 1 represents normal distribution with zero mean and standard deviation equal to 2 in all dimensions. The original datasets consist of 5000 instances each, 1000 of which have been selected for experiments.

11. **GLA**
    Glass type dataset, which is available at the UCI repository and represents the results of the forensic investigations aiming at determining the type of a glass based on a number of its physical properties and chemical content. At the scene of the crime, the broken glass left can be used as evidence if it is correctly identified.

12. **ION**
    Ionosphere radar data, which is available at the UCI repository and was collected by a system in Goose Bay, Labrador, consisting of a phased array of 16 high–frequency antennas with a total transmitted power on the order of 6.4 kilowatts. The targets were free electrons in the ionosphere. ”Good” radar returns are those showing evidence of some type of structure in the ionosphere. ”Bad” returns are those that do not; their signals pass through the ionosphere. Received signals were processed using an autocorrelation function whose arguments are the time of a pulse and the pulse number. There were 17 pulse numbers for the Goose Bay system. Instances in this database are described by 2 attributes per pulse number, corresponding to the complex values returned by the function resulting from the complex electromagnetic signal.

13. **IRI**
    The Iris plants database, which is available at the UCI repository and was originally created by R.A. Fisher. This is perhaps the best known dataset found in pattern recognition literature. The dataset contains 3 classes of 50 instances each, with 4 numeric attributes. Each class refers to a type of iris plant, while the attributes are simply the measures of length and width of the sepals and petals of iris. One class is linearly separable from the other two.

14. **LET**
    Letter image recognition dataset, which is available at the UCI repository. The objective is to identify each of a large number of black–and–white rectangular pixel displays as one

of the 26 capital letters in the English alphabet. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of unique stimuli. Each stimulus was converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 through 15. The original dataset consists of 20000 instances, 1000 of which have been selected for experiments.

15. **LIV**
    Liver disorder dataset, which is available at the UCI repository, trying to provide indication of liver disorder based on a number of attributes. The first 5 variables are all blood tests which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption. The sixth attribute gives the average alcoholic consumption given in the number of half–pint beverages drunk per day.

16. **PHO**
    Phoneme speech recognition dataset, which is available at the ELENA database and concerns French and Spanish speech recognition. The aim of this dataset is to distinguish between nasal (class 0) and oral (class 1) vowels. This dataset contains vowels coming from 1809 isolated syllables (for example: pa, ta, pan,...). Five different attributes were chosen to characterise each vowel: they are the amplitudes of the five first harmonics AHi, normalised by the total energy Ene (integrated on all the frequencies): AHi/Ene. Each harmonic is signed: positive when it corresponds to a local maximum of the spectrum and negative otherwise. The original dataset consists of 5404 instances, 1000 of which have been selected for experiments.

17. **SAT**
    The satellite images database, which is available at the UCI repository. This dataset was generated from Landsat Multi–Spectral Scanner image data. Each line of data corresponds to a 3x3 square neighbourhood of pixels completely contained within the 82x100 subarea. Each line contains the pixel values in the four spectral bands (converted to ASCII) of each of the 9 pixels in the 3x3 neighbourhood and a number indicating the classification label of the central pixel. The aim is to predict this classification, given the multi–spectral values. Seven classes of the type of a land are considered: red soil, cotton crop, grey soil, damp grey soil, soil with vegetation, stubble, mixture class (all types present), very damp grey soil, however no examples of mixture class were present. The original dataset consists of 6435 instances, 1000 of which have been selected for experiments.

18. **SEG**
    Image segmentation dataset, which is available at the UCI repository. The instances were drawn randomly from a database of 7 outdoor images. The images were hand–segmented to create a classification for every pixel. Each instance is a 3x3 region. The objective is to recognise one of the classes: brickface, sky, foliage, cement, window, path, grass, based on 19 typical image processing attributes. The original dataset consists of 2310 instances, 1000 of which have been selected for experiments.

19. **SHU**
    The shuttle dataset, which is available at the UCI repository and concerns automatic shuttle control. The dataset contains 9 attributes all of which are numerical. Among 7 classes corresponding to control actions the first class is dominant and accounts for 80%

of instances. The original dataset consists of 58000 instances, 1000 of which have been selected for experiments.

20. **SON**
    The sonar dataset, which is available at the UCI repository. The task is to train a network to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock. The dataset contains signals obtained from a variety of different aspect angles, spanning 90 degrees for the cylinder and 180 degrees for the rock. Each pattern is a set of 60 numbers in the range $[0, +1]$. Each number represents the energy within a particular frequency band, integrated over a certain period of time.

21. **SYN**
    Synthetic dataset, which is an artificial 2–class and 2–features dataset first used in [134]. Each of the partially overlapping classes compose of 2 Gaussians with shifted centres. The dataset comes divided into two non–overlapping parts (training and test) with 250 and 1000 instances respectively.

22. **TEX**
    The texture dataset, which is available at the ELENA database and concerns a study of textures discrimination with high order statistics. The aim is to distinguish between 11 different textures (Grass lawn, Pressed calf leather, Handmade paper, Raffia looped to a high pile, Cotton canvas, ...), each pattern being characterised by 40 attributes built by the estimation of fourth order modified moments in four orientations: 0, 45, 90 and 135 degrees. The original dataset consists of 5500 instances, 1000 of which have been selected for experiments.

23. **THY**
    Thyroid gland data, which is available at the UCI repository and is used for diagnostics, trying to predict whether a patient's thyroid could be classified as euthyroidism, hypothyroidism or hyperthyroidism. The diagnosis (class label) was based on a complete medical record, including anamnesis, scan etc.

24. **VEH**
    Vehicle silhouettes dataset, which is available at the UCI repository. The purpose of this dataset is to classify a given silhouette as one of four types of vehicle, using a set of features extracted from the silhouette. The vehicle may be viewed from one of many different angles. Four different vehicles (Opel, Saab, Bus, Van) were rotated and their angle of orientation was measured using a radial graticule beneath the vehicle. 0 and 180 degrees corresponded to 'head on' and 'rear' views respectively while 90 and 270 corresponded to profiles in opposite directions. Two sets of 60 images, each set covering a full 360 degree rotation, were captured for each vehicle rotated by a fixed angle between images.

25. **WIN**
    Wine recognition dataset, which is available at the UCI repository and provides data being the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents (features) found in each of the three types of wines (classes).

Appendix B

Classifiers, regressors and preprocessing techniques

B.1 Classifiers

A summary of classifiers used in the experiments described in this thesis has been given in Table B.1. Most of the classifiers are a part of the PRTools Pattern Recognition Toolbox for MATLAB [42]. Additional models based on physical fields described and derived in this work have been integrated with PRTools for compatibility and ease of use. A more detailed description of all classifiers is given below.

1. **FISHERC**
   Fisher's Least Square Linear Classifier [40], which finds the linear discriminant function between the classes in the dataset by minimising the errors in the least square sense. For multi–class situations the one–against–all strategy is used. For high–dimensional datasets or small sample sizes, the Pseudo–Fisher procedure based on a pseudo–inverse is used.

2. **LDC**
   Linear Bayes Normal Classifier [98], which computes the linear classifier between the classes of the dataset by assuming normal distributions with equal covariance matrices. The joint covariance matrix is the weighted (by a priori probabilities) average of the class covariance matrices.

3. **LOGLC**
   Logistic Linear Classifier [180], which computes the linear classifier for the dataset by maximising the likelihood criterion using the logistic (sigmoid) function.

4. **NMC**
   Nearest Mean Classifier, which assigns labels on the basis of proximity to class centroid. NMC is feature scaling sensitive and unsensitive to class priors.

5. **NMSC**
   Nearest Mean Scaled Classifier, which is based on an assumption of normal distributions and thereby automatically scales the features and is sensitive to class priors.

6. **QUADRC**
   Quadratic Discriminant Classifier [40], which computes the quadratic classifier between

the classes of the dataset assuming normal distributions and is a quadratic equivalent of FISHERC. The multi–class problem is solved by multiple two–class quadratic discriminants.

7. **QDC**
   Quadratic Bayes Normal Classifier [98], which computes the quadratic classifier between the classes of the dataset assuming normal distributions and is a quadratic equivalent of LDC. Unlike QUADRC, this routine uses the densities rather than class covariances.

8. **UDC**
   Uncorrelated Normal Based Quadratic Bayes Classifier [40], which computes a quadratic classifier between the classes in the dataset assuming normal distributions with uncorrelated features (diagonal covariance matrices). The classifier is in fact a quadratic version of the Naive Bayes classifier described below.

9. **KLLDC**
   Linear Classifier built on the Karhunen–Loeve expansion of the common covariance matrix [40, 98], which finds the linear discriminant function for a dataset by computing the LDC on the data projected on the eigenvectors of the averaged covariance matrix of the classes.

10. **PCLDC**
    Linear Classifier using Principal Component expansion on the joint data [40, 98], which finds the linear discriminant function for a dataset by computing the LDC on a projection of the data on the eigenvectors of the covariance matrix for the total dataset.

11. **KNNC**
    k–Nearest Neighbour Classifier [40, 98] with automatic selection of the nearest neighbour count by optimisation with respect to the leave–one–out error on the dataset.

12. **PARZENC**
    Parzen Density Based Classifier [98] with automatic selection of the smoothing parameter value by optimisation with respect to the leave–one–out error on the dataset.

13. **TREEC**
    Decision Tree Classifier [40, 98].

14. **NAIVEBC**
    Naive Bayes Classifier [40], which estimates the density for every class and every feature separately. Total class densities are constructed by assuming independency and consequently multiplying the separate feature densities.

15. **PERLC**
    Linear Perceptron Classifier [11, 40, 98], which is in fact a feed–forward neural network without hidden layers and with linear units.

16. **RBNC**
    Radial Basis Function Neural Network Classifier [11], which is in fact a feed–forward neural network with one hidden layer with radial basis units.

17. **SVC**
    Support Vector Classifier based on C–SVM [40, 176] with linear kernels (default) and an error penalty parameter C, which controls solution complexity.

18. **NUSVC**
    Support Vector Classifier based on ν–SVM [176] with linear kernels (default) and an upper bound parameter ν for the expected classification error, which can be estimated automatically by the leave–one–out error.

19. **GFC**
    Gravity Field Classifier, see Section 3.2.1 for details.

20. **EFC**
    Electrostatic Field Classifier, see Section 3.2.2 for details.

Table B.1: Classifiers used in the experiments

no.	acronym	source	description
1	fisherc	PRTools	Fisher's Linear Classifier
2	ldc	PRTools	Linear Bayes Normal Classifier
3	loglc	PRTools	Logistic Linear Classifier
4	nmc	PRTools	Nearest Mean Classifier
5	nmsc	PRTools	Nearest Mean Scaled Classifier
6	quadrc	PRTools	Quadratic Discriminant Classifier
7	qdc	PRTools	Quadratic Bayes Normal Classifier
8	udc	PRTools	Uncorrelated Quadratic Bayes Normal Classifier
9	klldc	PRTools	Linear Classifier using Karhunen–Loeve expansion
10	pcldc	PRTools	Linear Classifier using Principal Component expansion
11	knnc	PRTools	k–Nearest Neighbor Classifier
12	parzenc	PRTools	Parzen Density Classifier
13	treec	PRTools	Decision Tree Classifier
14	naivebc	PRTools	Naive Bayes Classifier
15	perlc	PRTools	Linear Perceptron Classifier
16	rbnc	PRTools	RBF Neural Network Classifier
17	svc	PRTools	Support Vector Machine classifier (C–SVM)
18	nusvc	PRTools	Support Vector Machine classifier (ν–SVM)
19	gfc	thesis	Gravity Field Classifier
20	efc	thesis	Electrostatic Field Classifier
21	efc–ldr	thesis	EFC with Local Dimensionality Reduction
22	efc–cr	thesis	EFC with Charge Redistribution
23	efc–mimp	thesis	EFC with Class–conditional Mean Imputation
24	efc–cwd	thesis	EFC with Casewise Deletion
25	efc–gfc	thesis	EFC with GFC–fallback
26	efc–pc	thesis	EFC with Pre–Classification

B.2 Regressors

A summary of regression techniques (regressors) used in the experiments described in this thesis has been given in Table B.2. Most of the regressors are a part of the PRTools Toolbox. Additional models used in this work have been integrated with PRTools. A more detailed description of all regressors is given below.

1. **LASSOR**
   Least Absolute Shrinkage and Selection Operator Linear Regression [163], which minimises the sum of squared errors with a bound on the sum of the absolute values of the coefficients. Because of this constraint, the method tends to produce some coefficients which are exactly 0, acting as a feature selection algorithm enabling interpretation of the generated models. LASSO regression also exhibits stability similar to ridge regression described below.

2. **LINEARR**
   Ordinary Least Squares (OLS) Linear Regression [176].

3. **KNNR**
   k–Nearest Neighbour Regression, which calculates predictions by averaging the target variable values of k nearest neighbours in the input instance.

4. **GRNNR**
   Generalised Regression Network Regression, based on MATLAB's implementation of a Radial Basis Function (RBF) network [11].

5. **PLSR**
   Partial Least Squares Regression [182], which finds a linear regression model by projecting the target and input variables to a new space. This new space is constructed to explain as much as possible of the covariance between inputs and targets.

6. **RIDGER**
   Ridge Regression [164], described in more detail in Section 6.2.6.

7. **BOOSTR**
   Regression Boost, which is a weighted combination of multiple instances of a base regressor, trained on various versions of the input dataset. The method has been developed for use in the Toxicity Prediction Challenge described in Section 6.2 and has been inspired by the AdaBoost algorithm for classification problems [52]. At each iteration a single base regressor is trained using data sampled from the original input dataset $\mathcal{D}$ according to initially uniform distribution W. Thus denoting by $W(i)$ the probability of selecting the i^{th} out of N instances for training, initially $\forall_i W(i) = 1/N$. In consecutive iterations the distribution is updated according to: $W(i) \leftarrow W(i)(1 + err(i))/z$, where $err(i)$ is the absolute error for the i^{th} instance and z is a scaling factor used to ensure that W remains a true distribution. After a given number of base regressors has been trained, the predictions of the final ensemble become averaged (median) predictions of the members weighted by an exponential function of their normalised Mean Absolute Error on $\mathcal{D}$.

8. **MLPR**
   Cross–trained Multi–Layer Perceptron (MLP) Ensemble Regression [145], based on

MATLAB's implementation of a Multi–Layer Perceptron and median combination of ensemble members' predictions.

Table B.2: Regressors used in the experiments

no.	acronym	source	description
1	lassor	PRTools	Least Absolute Shrinkage and Selection Operator linear regression
2	linearr	PRTools	Ordinary Least Squares (OLS) linear regression
3	knnr	PRTools	k–Nearest Neighbour regression
4	grnnr	MATLAB	Generalised Regression Network regression
5	plsr	PRTools	Partial Least Squares regression
6	ridger	PRTools	Ridge regression
7	boostr	thesis	Regression Boost
8	mlpr	thesis	Cross–trained Multi–Layer Perceptron ensemble regression

B.3 Preprocessors

A summary of preprocessing techniques used in the experiments described in this thesis has been given in Table B.3. Most of the preprocessors are a part of the PRTools Toolbox. Additional methods used in this work have been integrated with PRTools. A more detailed description of all preprocessors is given below.

1. **FEATSELM**
   Greedy feature selection based on a cross–validation criterion [175]. The following selection algorithms are included:

   - **FEATSELF** – forward feature selection, which starts with an empty feature set and adds a single locally best feature at each iteration, until a prescribed number of features is obtained.
   - **FEATSELB** – backward feature selection, which starts with complete feature set and removes a single locally worst feature at each iteration, until a prescribed number of features remains.
   - **FEATSELLR** – plus–L–takeaway–R feature selection, which is a combination of the forward and backward method, making L forward steps followed by R backward steps at each iteration.

2. **PCA**
   Principal Component Analysis [87], which is a procedure for transformation of a number of possibly correlated variables into a smaller number of uncorrelated variables called Principal Components (PCs). The first principal component accounts for as much of the variability in the data as possible, and each consecutive component accounts for as much of the remaining variability as possible. PCA is thus a procedure for reduction of dataset dimensionality preserving the maximum level of variance, without taking advantage of class information given with the data (unsupervised procedure).

3. **ROBPCA**
   Robust PCA [178] is a resistant to outliers version of PCA, which uses projection–pursuit techniques and the Minimum Covariance Determinant method.

4. **FISHERM**
   Linear Discriminant Analysis [40], which is a method that finds a linear projection of the data which best separates the classes and is thus useful for discrimination purposes. The shortcoming of LDA is that the maximum dimensionality of the projection is limited to $(c-1)$, where c is the number of classes [40]. Unlike PCA though, LDA tries to take advantage of the class information, so better classification performance can be expected.

5. **CCIMPC**
   Class–conditional Imputation of mean or maximum likelihood estimate from a distribution assuming independence of features. Both methods have been described in more detail in Sections 3.4.2 and 6.3.5.

6. **PBIL**
   Population Based Incremental Learning [8], which is a semi–random feature selection method and combines the mechanisms of a generational genetic algorithm with competitive learning. PBIL replaces the population with a single probability vector from which samples can be drawn to produce the next generation's population. The probability vector is then updated on the basis of performance of the population and the whole procedure is repeated.

Table B.3: Preprocessing techniques used in the experiments

no.	acronym	source	description
1	featselm	PRTools	Greedy feature selection (forward, backward and combined)
2	pca	PRTools	Principal Component Analysis
3	robpca	LIBRA	Robust Principal Component Analysis
4	fisherm	PRTools	Linear Discriminant Analysis
5	ccimpc	thesis	Class–conditional imputation of mean or ML estimate
6	pbil	thesis	Population Based Incremental Learning

Appendix C

Information Theoretic Learning framework

C.1 Renyi's quadratic entropy

The definition of Renyi's entropy of order α for a discrete random variable is given by [127]:

$$H_{R_\alpha}(Y) = \frac{1}{1-\alpha} \log \sum_{k=1}^{n} P(Y_k)^\alpha \ , \quad \alpha > 0, \ \alpha \neq 1 \tag{C.1}$$

By analogy to Shannon's differential entropy H_S, Renyi's entropy of order α for a continuous random variable is:

$$H_{R_\alpha}(Y) = \frac{1}{1-\alpha} \log \int p(y)^\alpha \ dy \tag{C.2}$$

Renyi's entropy involves calculation of the integral of the power of PDF rather than integral of the logarithm as in the case of Shannon's counterpart, which is much easier to estimate [128]. Moreover, Shannon's entropy is the limiting case of Renyi's entropy when $\alpha \to 1$. For practical applications the choice of $\alpha = 2$ is a good compromise between robustness and computational complexity $O(n^2)$ [128], which leads to the definition of Renyi's quadratic entropy:

$$H_{R_2}(Y) = -\log \int p(y)^2 \ dy \tag{C.3}$$

The important property of Renyi's entropy from the point of view of ITL is that the extrema of H_S and H_R overlap [127], so both definitions are equivalent for the purpose of entropy optimisation.

For the above criterion to be useful, an estimate of the probability density function is needed. Fortunately, the well known Parzen window density estimator [40] can be easily and efficiently integrated into the formula for calculation of Renyi's entropy. Denoting by $G(\boldsymbol{y}, \ \sigma^2 \boldsymbol{I})$ a spherical Gaussian kernel centered at $\boldsymbol{y}$ with a diagonal covariance matrix $\sigma^2 \boldsymbol{I}$, and by N the number of available instances of Y, the PDF can be estimated as follows:

$$p(\boldsymbol{y}) = \frac{1}{N} \sum_{i=1}^{N} G(\boldsymbol{y} - \boldsymbol{y}_i, \ \sigma^2 \boldsymbol{I}) \tag{C.4}$$

Substituting Eq. C.4 into Eq. C.3 and using the Gaussian convolution property:

$$H_{R_2}(Y) = -\log \int p(\boldsymbol{y})^2 \, d\boldsymbol{y} = -\log V(\boldsymbol{y}) \tag{C.5}$$

where

$$V(\boldsymbol{y}) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \int G(\boldsymbol{z} - \boldsymbol{y}_i, \, \sigma^2 \boldsymbol{I}) \, G(\boldsymbol{z} - \boldsymbol{y}_j, \, \sigma^2 \boldsymbol{I}) \, d\boldsymbol{z} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} G(\boldsymbol{y}_i - \boldsymbol{y}_j, \, 2\sigma^2 \boldsymbol{I})$$

Since the logarithm is a monotonic function, optimisation of H_{R_2} is equivalent to optimisation of $V(\boldsymbol{y})$. This however is not the main reason why Eq. C.5 has been written in this way. If some imaginary physical particles were placed on top of each data point $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ a potential field would be created, since $G(\boldsymbol{y}_i - \boldsymbol{y}_j, \; 2\sigma^2 \boldsymbol{I})$ is always positive and decays exponentially with the square of the distance between $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ [127]. For this reason the data instances can be called information particles and $V(\boldsymbol{y})$, which is an averaged sum of all pairs of interactions – the Information Potential. In general, Renyi's entropy of order α calculates the interactions between $\alpha-$tuplets of instances, so the higher the value of α, the more information about the structure of the dataset can be extracted [128], but the computational complexity of $O(N^\alpha)$quickly becomes prohibitive.

The notion of Information Potential leads to another concept – Information Force (IF) exerted on each information particle by the field. By analogy to classical physics, the IF is given by:

$$\frac{\partial G(\boldsymbol{y}_i - \boldsymbol{y}_j, \; 2\sigma^2 \boldsymbol{I})}{\partial \boldsymbol{y}_i} = -G(\boldsymbol{y}_i - \boldsymbol{y}_j, \; 2\sigma^2 \boldsymbol{I}) \frac{\boldsymbol{y}_i - \boldsymbol{y}_j}{2\sigma^2} \tag{C.6}$$

which leads to the following equation for the total force exerted on instance y_i by all other Information Particles:

$$\boldsymbol{F}_i = \frac{\partial V(\boldsymbol{y})}{\partial \boldsymbol{y}_i} = \frac{-1}{N^2 \sigma^2} \sum_{j=1}^{N} G(\boldsymbol{y}_i - \boldsymbol{y}_j, \; 2\sigma^2 \boldsymbol{I})(\boldsymbol{y}_i - \boldsymbol{y}_j) \tag{C.7}$$

If the Information Particles were free to move in the potential field, the forces would drive them all to a state with minimum potential. This behaviour can be immediately used for training of various adaptive systems such as Multi–Layer Perceptrons by backpropagating the forces, which as a result take place of the injected error [128].

Since the value of the Parzen window width parameter σ determines the range of particle interactions and all particles should be able to interact with each other, σ must be set accordingly. According to [169] a good practical solution is to set σ to about half of the largest distance between data instances and slowly decrease its value over time (annealing). This facilitates finding an acceptable global solution in the initial phase of training and then fine–tuning it, using more and more localized interactions.

C.2 Mutual information between continuous random variables

Estimation of mutual information from a set of instances is more difficult than estimation of the entropy. Mutual information between two random variables is equal to Kullback–Leibler divergence between the joint PDF and a product of marginal PDFs. As discussed in Chapter 5,

Kullback–Leibler divergence is not amenable to non–parametric estimation, especially in higher dimensions. The Renyi's divergence measure of order α, which in the limit of $\alpha \to 1$ becomes in fact the Kullback–Leibler divergence is not quadratic in the PDF even for $\alpha = 2$. Fortunately there exist other criteria that can be used.

For the case in which all random variables are continuous (there can be more than two variables involved), two different mutual information estimators have been proposed in [127]. Both of them have the desired property of being quadratic functions of the PDFs. The definitions given below apply to the two–variable scenario and the extension to multiple variables will be given later:

- Cauchy–Schwarz quadratic MI, based on the Cauchy–Schwarz inequality:

$$I_{CS}(Y_1, Y_2) = \log \frac{\left(\iint p(\boldsymbol{y}_1, \boldsymbol{y}_2)^2 \, d\boldsymbol{y}_1 \, d\boldsymbol{y}_2\right) \left(\iint p(\boldsymbol{y}_1)^2 p(\boldsymbol{y}_2)^2 \, d\boldsymbol{y}_1 \, d\boldsymbol{y}_2\right)}{\left(\iint p(\boldsymbol{y}_1, \boldsymbol{y}_2) p(\boldsymbol{y}_1) p(\boldsymbol{y}_2) \, d\boldsymbol{y}_1 \, d\boldsymbol{y}_2\right)^2} \tag{C.8}$$

- Euclidean distance quadratic MI, based on the Euclidean difference of vectors inequality:

$$\begin{aligned} I_{ED}(Y_1, Y_2) &= \iint p(\boldsymbol{y}_1, \boldsymbol{y}_2)^2 \, d\boldsymbol{y}_1 \, d\boldsymbol{y}_2 + \iint p(\boldsymbol{y}_1)^2 p(\boldsymbol{y}_2)^2 \, d\boldsymbol{y}_1 \, d\boldsymbol{y}_2 \\ &- 2 \iint p(\boldsymbol{y}_1, \boldsymbol{y}_2) p(\boldsymbol{y}_1) p(\boldsymbol{y}_2) \, d\boldsymbol{y}_1 \, d\boldsymbol{y}_2 \end{aligned} \tag{C.9}$$

According to experimental results given in [127] both I_{CS} and I_{ED} behave in a similar way when it comes to minimisation of MI, but I_{ED} is better suited for MI maximisation. For more principled treatment of suitability of I_{ED} for optimisation of MI see [169] and references therein. Note, that from Eqs. C.8 and C.9 it can be seen that both formulas are always non–negative and equal 0 only when Y_1 and Y_2 are independent, which makes them valid divergence measures.

Using the Parzen window formula of Eq. C.4, the joint and marginal probability density functions for the case of k continuous random variables, can be estimated as follows:

$$p(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k) = \frac{1}{N} \sum_{i=1}^{N} \prod_{l=1}^{k} G(\boldsymbol{y}_l - \boldsymbol{y}_{li}, \, \sigma_l^2 \boldsymbol{I}) \tag{C.10}$$

$$p(\boldsymbol{y}_l) = \frac{1}{N} \sum_{i=1}^{N} G(\boldsymbol{y}_l - \boldsymbol{y}_{li}, \, \sigma_l^2 \boldsymbol{I}) \tag{C.11}$$

Notice, that unlike the original formulas given in [127], Eqs. C.10 and C.11 allow each marginal distribution to have its own Gaussian kernel width, which allows for greater flexibility, especially in the case of optimisation of MI between input and output (switch in position 2 in Figure 2.10).

If the equations above are combined with Eqs. C.8 and C.9, the following formulas to estimate the mutual information from a dataset are produced:

$$I_{CS}(Y_1, \ldots, Y_k) = \log \frac{V(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k) \prod_{l=1}^{k} V(\boldsymbol{y}_l)}{V_{nc}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k)^2} \tag{C.12}$$

$$I_{ED}(Y_1, \ldots, Y_k) = V(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k) + \prod_{l=1}^{k} V(\boldsymbol{y}_l) - 2 V_{nc}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k) \tag{C.13}$$

where:

$$V(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_k) = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\prod_{l=1}^{k} G(\boldsymbol{y}_{li}-\boldsymbol{y}_{lj},\ 2\sigma_l^2\boldsymbol{I})$$

$$V_j(\boldsymbol{y}_l) = \frac{1}{N}\sum_{i=1}^{N} G(\boldsymbol{y}_{lj}-\boldsymbol{y}_{li},\ 2\sigma_l^2\boldsymbol{I})$$

$$V(\boldsymbol{y}_l) = \frac{1}{N}\sum_{j=1}^{N} V_j(\boldsymbol{y}_l)$$

$$V_{nc}(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_k) = \frac{1}{N}\sum_{j=1}^{N}\prod_{l=1}^{k} V_j(\boldsymbol{y}_l)$$

The above equations can be interpreted in terms of the information potentials:

- $V(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_k)$ is called the Joint Information Potential (JIP) as it is defined in the joint space of all random variables involved,
- $V_j(\boldsymbol{y}_l)$ is a Partial Marginal Information Potential (PMIP) as it is the potential of instance $\boldsymbol{y}_{lj}$ in its corresponding marginal potential field; note that it is equal to the Information Potential given in Section C.1,
- $V(\boldsymbol{y}_l)$ is the Marginal Information Potential (MIP), which averages all Partial Marginal Information Potentials for a single random variable,
- $V_{nc}(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_k)$ is the Unnormalised Cross Information Potential (UCIP), which measures the interactions between Partial Marginal Information Potentials.

If the nominator of the logarithm in Eq. C.12 is treated as a normalisation term for $V_{nc}(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_k)$, normalised Cross Information Potential (CIP) can be defined as follows:

$$V_c(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_k) = \frac{V_{nc}(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_k)^2}{V(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_k)\prod_{l=1}^{k} V(\boldsymbol{y}_l)} \tag{C.14}$$

Note, that by using the CIP formula, the estimator of I_{CS} becomes analytically pleasing due to consistency with the equation for calculation of Renyi's quadratic entropy:

$$\begin{aligned} H_{R_2}(\boldsymbol{y}) &= -\log V(\boldsymbol{y}) \\ I_{CS}(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_k) &= -\log V_c(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_k) \end{aligned}$$

Unlike in the case of entropy optimisation, now the force exerted on every information particle called the Marginal Information Force (MIF), comes from three different and independent sources. The total marginal force exerted on instance $\boldsymbol{y}_{li}$ is given by:

$$\begin{aligned} \frac{\partial I_{CS}(Y_1,\ldots,Y_k)}{\partial \boldsymbol{y}_{li}} &= \frac{1}{V(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_k)}\frac{\partial V(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_k)}{\partial \boldsymbol{y}_{li}} + \frac{1}{V(\boldsymbol{y}_l)}\frac{\partial V(\boldsymbol{y}_l)}{\partial \boldsymbol{y}_{li}} \\ &- 2\left(\frac{1}{V_{nc}(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_k)}\frac{\partial V_{nc}(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_k)}{\partial \boldsymbol{y}_{li}}\right) \end{aligned} \tag{C.15}$$

$$\frac{\partial I_{ED}(Y_1,\ldots,Y_k)}{\partial \boldsymbol{y}_{li}} = \frac{\partial V(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_k)}{\partial \boldsymbol{y}_{li}} + \left(\prod_{m\neq l} V(\boldsymbol{y}_m)\right)\frac{\partial V(\boldsymbol{y}_l)}{\partial \boldsymbol{y}_{li}} - 2\left(\frac{\partial V_{nc}(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_k)}{\partial \boldsymbol{y}_{li}}\right) \tag{C.16}$$

where:

$$\begin{aligned}
\frac{\partial V(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_k)}{\partial \boldsymbol{y}_{li}} &= \frac{-1}{N^2\sigma_l^2}\sum_{j=1}^{N}\left(\prod_{m=1}^{k} G(\boldsymbol{y}_{mi}-\boldsymbol{y}_{mj},\ 2\sigma_m^2\boldsymbol{I})\right)(\boldsymbol{y}_{li}-\boldsymbol{y}_{lj}) \\
\frac{\partial V(\boldsymbol{y}_l)}{\partial \boldsymbol{y}_{li}} &= \frac{-1}{N^2\sigma_l^2}\sum_{j=1}^{N} G(\boldsymbol{y}_{li}-\boldsymbol{y}_{lj},\ 2\sigma_l^2\boldsymbol{I})\,(\boldsymbol{y}_{li}-\boldsymbol{y}_{lj}) \\
\frac{\partial V_{nc}(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_k)}{\partial \boldsymbol{y}_{li}} &= \frac{-1}{N^2\sigma_l^2}\sum_{j=1}^{N}\left(\prod_{m\neq l} V_j(\boldsymbol{y}_m)\right) G(\boldsymbol{y}_{li}-\boldsymbol{y}_{lj},\ 2\sigma_l^2\boldsymbol{I})\,(\boldsymbol{y}_{li}-\boldsymbol{y}_{lj})
\end{aligned}$$

Note, that in Eq. C.15 the forces are normalised by corresponding potentials, which is a consequence of the logarithm in the definition of I_{CS} [127].

C.3 Mutual information between continuous and discrete random variables

The extension of mutual information estimators presented to a discrete variable case has been first given in [170] and then used multiple times, e.g. in [166, 167, 168, 169, 165]. The idea was to design a supervised feature extractor for classification and class visualisation, using MI as the criterion to maximise. In this scenario, the desired signal is the class label and can be thought of as a discrete random variable. Since, as said before, I_{ED} is better behaved in the case of maximisation of mutual information, only this estimator has been extended to the discrete case. Denoting by C the class label variable, the MI estimate is given by:

$$I_{ED}(C,Y) = \sum_C \int p(c,\boldsymbol{y})^2 d\boldsymbol{y} + \sum_C \int p(c)^2 p(\boldsymbol{y})^2 d\boldsymbol{y} - 2\sum_C \int p(c,\boldsymbol{y})p(c)p(\boldsymbol{y})d\boldsymbol{y} \tag{C.17}$$

Let J_p denote the number of instances of class c_p and N_c the number of classes. Since $\sum_{p=1}^{N_c} J_p = N$, class prior probabilities are given by $P(c_p) = J_p/N$. The Parzen density estimate for each class c_p can now be calculated as ($\boldsymbol{y}_{pj}$ denotes j^{th} instance within p^{th} class):

$$p(\boldsymbol{y}|c_p) = \frac{1}{J_p}\sum_{j=1}^{J_p} G\left(\boldsymbol{y}-\boldsymbol{y}_{pj},\sigma^2\boldsymbol{I}\right) \tag{C.18}$$

Applying the definition of joint density $p(c,\boldsymbol{y}) = p(\boldsymbol{y}|c)P(c)$, the density estimate becomes:

$$p(c_p,\boldsymbol{y}) = \frac{1}{N}\sum_{j=1}^{J_p} G\left(\boldsymbol{y}-\boldsymbol{y}_{pj},\sigma^2\boldsymbol{I}\right) \tag{C.19}$$

Since for all data the density $p(\boldsymbol{y}) = \sum_C p(c, \boldsymbol{y})$:

$$p(\boldsymbol{y}) = \frac{1}{N} \sum_{p=1}^{N_c} \sum_{j=1}^{J_p} G\left(\boldsymbol{y} - \boldsymbol{y}_{pj}, \sigma^2 \boldsymbol{I}\right) = \frac{1}{N} \sum_{i=1}^{N} G\left(\boldsymbol{y} - \boldsymbol{y}_i, \sigma^2 \boldsymbol{I}\right) \tag{C.20}$$

Using the above, the components of $I_{ED}(C, Y)$ become:

$$V_{(cy)^2} = \sum_C \int p(c, \boldsymbol{y})^2 d\boldsymbol{y} = \frac{1}{N^2} \sum_{p=1}^{N_c} \sum_{k=1}^{J_p} \sum_{l=1}^{J_p} G\left(\boldsymbol{y}_{pk} - \boldsymbol{y}_{pl}, 2\sigma^2 \boldsymbol{I}\right) \tag{C.21}$$

$$V_{c^2 y^2} = \sum_C \int P(c)^2 p(\boldsymbol{y})^2 d\boldsymbol{y} = \frac{1}{N^2} \left(\sum_{p=1}^{N_c} \left(\frac{J_p}{N} \right)^2 \right) \sum_{k=1}^{N} \sum_{l=1}^{N} G\left(\boldsymbol{y}_k - \boldsymbol{y}_l, 2\sigma^2 \boldsymbol{I}\right) \tag{C.22}$$

$$V_{cy} = \sum_C \int p(c, \boldsymbol{y}) P(c) p(\boldsymbol{y}) d\boldsymbol{y} = \frac{1}{N^2} \sum_{p=1}^{N_c} \frac{J_p}{N} \sum_{j=1}^{J_p} \sum_{k=1}^{N} G\left(\boldsymbol{y}_{pj} - \boldsymbol{y}_k, 2\sigma^2 \boldsymbol{I}\right) \tag{C.23}$$

The above quantities can be considered as information potentials and interpreted as:

- $V_{(cy)^2}$ – interactions between pairs of particles inside each class, summed over all classes,
- $V_{c^2 y^2}$ – interactions between all pairs of particles regardless of class, weighted by the sum of squared class priors,
- V_{cy} – interactions between instances of a particular class against all instances, weighted by the class prior and summed over all classes.

The information forces representing the directions and magnitudes where the transform would move the particles in order to maximise MI can be calculated as:

$$\frac{\partial V_{(cy)^2}}{\partial \boldsymbol{y}_{ci}} = \frac{1}{N^2 \sigma^2} \sum_{k=1}^{J_c} G\left(\boldsymbol{y}_{ck} - \boldsymbol{y}_{ci}, 2\sigma^2 \boldsymbol{I}\right) \left(\boldsymbol{y}_{ck} - \boldsymbol{y}_{ci}\right) \tag{C.24}$$

$$\frac{\partial V_{c^2 y^2}}{\partial \boldsymbol{y}_{ci}} = \frac{1}{N^2 \sigma^2} \left(\sum_{p=1}^{N_c} \left(\frac{J_p}{N} \right)^2 \right) \sum_{k=1}^{N} G\left(\boldsymbol{y}_k - \boldsymbol{y}_i, 2\sigma^2 \boldsymbol{I}\right) \left(\boldsymbol{y}_k - \boldsymbol{y}_i\right) \tag{C.25}$$

$$\frac{\partial V_{cy}}{\partial \boldsymbol{y}_{ci}} = \frac{1}{N^2 \sigma^2} \sum_{p=1}^{N_c} \frac{J_p + J_c}{2N} \sum_{j=1}^{J_c} G\left(\boldsymbol{y}_{pj} - \boldsymbol{y}_{ci}, 2\sigma^2 \boldsymbol{I}\right) \left(\boldsymbol{y}_{pj} - \boldsymbol{y}_{ci}\right) \tag{C.26}$$

Interpretation of the forces is as follows [170]:

- $\frac{\partial V_{(cy)^2}}{\partial y_{ci}}$ – sum of forces that other particles in class c exert on particle $\boldsymbol{y}_{ci}$ (the direction is towards $\boldsymbol{y}_{ci}$),
- $\frac{\partial V_{c^2 y^2}}{\partial y_{ci}}$ – sum of forces that other particles regardless of class exert on particle $\boldsymbol{y}_{ci}$ (denoted by $\boldsymbol{y}_i$ when the class is irrelevant, the direction is towards $\boldsymbol{y}_i$),
- $\frac{\partial V_{cy}}{\partial y_{ci}}$ – repulsion of classes away from each other.

References

[1] C. Aggarwal, "Re–designing distance functions and distance–based applications for high dimensional data", *ACM SIGMOD Record*, vol. 30, no. 1, pp. 13–18, 2001.

[2] C. Aggarwal, A. Hinneburg, and D. Keim, "On the surprising behavior of distance metrics in high dimensional space", in *Database Theory – ICDT 2001*, ser. Lecture Notes in Computer Science, J. Bussche and V. Vianu, Eds. Springer, 2001, vol. 1973, pp. 420–435.

[3] P. Aguilera, A. Frenich, J. Torres, H. Castro, J. Vidal, and M. Canton, "Application of the Kohonen neural network in coastal water management: methodological development for the assessment and prediction of water quality", *Water Research*, vol. 35, no. 17, pp. 4053–4062, 2001.

[4] E. Alpaydin, *Introduction to machine learning*. Cambridge, USA: The MIT Press, 2004.

[5] A. Antos, L. Devroye, and L. Gyorfi, "Lower bounds for Bayes error estimation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 7, pp. 643–645, 1999.

[6] A. Asuncion and D. Newman, "UCI Machine Learning Repository", 2007. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[7] P. Ball, *Critical mass: How one thing leads to another*. London, UK: Arrow Books Ltd., 2005.

[8] S. Baluja, "Population–based incremental learning: A method for integrating genetic search based function optimization and competitive learning", Pittsburgh, PA: Carnegie Mellon University, Tech. Rep., 1994.

[9] R. Battiti, "Using mutual information for selecting features in supervised neural net learning", *IEEE Transactions on Neural Networks*, vol. 5, no. 4, pp. 537–550, 1994.

[10] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is "nearest neighbor" meaningful", in *Databases Theory – ICDT 1999*, ser. Lecture Notes in Computer Science, C. Beeri and P. Buneman, Eds. Springer, 1999, vol. 1540, pp. 217–235.

[11] C. Bishop, *Neural Networks for Pattern Recognition*. New York, USA: Oxford University Press, 1995.

[12] C. Bishop, *Pattern recognition and machine learning*. New York, USA: Springer, 2006.

[13] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co–training", in *Proceedings of the 11th annual conference on Computational learning theory*. ACM, 1998, pp. 92–100.

[14] R. Buccigrossi and E. Simoncelli, "Image compression via joint statistical characterization in the wavelet domain", *IEEE Transactions on Image Processing*, vol. 8, pp. 1688–1701, 1999.

[15] M. Budka and B. Gabrys, "Electrostatic Field Classifier for Deficient Data", in *Computer Recognition Systems 3*, ser. Advances in Soft Computing, M. Kurzynski and M. Wozniak, Eds. Springer, 2009, vol. 57, pp. 311–318.

[16] M. Budka and B. Gabrys, "Correntropy–based density–preserving data sampling as an alternative to standard cross–validation", in *Proceedings of the IEEE World Congress on Computational Intelligence*. IEEE, 2010, pp. 1437–1444.

[17] M. Budka and B. Gabrys, "Ridge regression ensemble for toxicity prediction", *Procedia Computer Science*, vol. 1, no. 1, pp. 193–201, 2010.

[18] M. Budka and B. Gabrys, "Density Preserving Sampling (DPS) for error estimation and model selection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010 (Submitted).

[19] M. Budka and B. Gabrys, "On accuracy of PDF divergence estimators and their applicability to representative data sampling", *IEEE Transactions on Information Theory*, 2010 (Submitted).

[20] M. Budka and B. Gabrys, "Electrostatic field framework for supervised and semi–supervised learning from incomplete data", *Natural Computing*, DOI:10.1007/s11047-010-9182-4.

[21] M. Budka, B. Gabrys, and E. Ravagnan, "Robust predictive modelling of water pollution using biomarker data", *Water Research*, vol. 44, no. 10, pp. 3294–3308, 2010.

[22] J. Cardoso, "Infomax and maximum likelihood for blind source separation", *IEEE Signal processing letters*, vol. 4, no. 4, pp. 112–114, 1997.

[23] J. Cardoso, "Blind signal separation: statistical principles", *Proceedings of the IEEE*, vol. 86, no. 10, pp. 2009–2025, 1998.

[24] V. Černỳ, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm", *Journal of optimization theory and applications*, vol. 45, no. 1, pp. 41–51, 1985.

[25] N. Chèvre, F. Gagné, P. Gagnon, and C. Blaise, "Application of rough sets analysis to identify polluted aquatic sites based on a battery of biomarkers: a comparison with classical methods", *Chemosphere*, vol. 51, no. 1, pp. 13–23, 2003.

[26] A. Cichocki and S. Amari, "Families of Alpha–Beta–and Gamma–Divergences: Flexible and Robust Measures of Similarities", *Entropy*, vol. 12, no. 6, pp. 1532–1568, 2010.

[27] P. Coles, T. Cox, C. Mackey, and S. Richardson, "The toxic terabyte", White paper, IBM Global Technology Services, 2006. [Online]. Available: http://www-03.ibm.com/systems/resources/systems_storage_solutions_pdf_toxic_tb.pdf

[28] C. Cortes and V. Vapnik, "Support–vector networks", *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[29] T. Cover, "Learning in Pattern Recognition." Stanford University CA, Stanford Electronics Labs, Tech. Rep., 1968.

[30] T. Cover and J. Thomas, *Elements of information theory*. New York, USA: John Wiley & Sons, 2006.

[31] N. Cristianini and J. Shawe-Taylor, *An introduction to support Vector Machines: and other kernel-based learning methods*. New York, USA: Cambridge University Press, 2000.

[32] R. Dara, S. Kremer, and D. Stacey, "Clustering unlabeled data with SOMs improves classification of labeled real–world data", in *Proceedings of the 2002 International Joint Conference on Neural Networks*, vol. 3. IEEE, 2002, pp. 2237–2242.

[33] K. Darlington, *The essence of expert systems*. New Jersey, USA: Prentice Hall, 2000.

[34] A. Dempster, N. Laird, and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm", *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[35] I. Dhillon, S. Mallela, and R. Kumar, "A divisive information theoretic feature clustering algorithm for text classification", *Journal of Machine Learning Research*, vol. 3, pp. 1265–1287, 2003.

[36] T. Dietterich, "Ensemble methods in machine learning", *Multiple classifier systems*, vol. 1857, pp. 1–15, 2000.

[37] Y. Dodge, D. Cox, D. Commenges, A. Davison, and P. Solomon, *The Oxford dictionary of statistical terms*. Oxford, UK: Oxford University Press, 2006.

[38] M. Dorigo and L. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem", *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[39] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents", *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 26, no. 1, pp. 29–41, 1996.

[40] R. Duda, P. Hart, and D. Stork, *Pattern Classification 2nd ed.* New York, USA: John Wiley & Sons, 2001.

[41] R. Duin, "On the choice of smoothing parameters for Parzen estimators of probability density functions", *IEEE Transactions on Computers*, vol. 100, no. 25, pp. 1175–1179, 1976.

[42] R. Duin, P. Juszczak, P. Paclik, E. Pekalska, D. de Ridder, D. Tax, and S. Verzakov, "PR–Tools 4.1, A MATLAB Toolbox for Pattern Recognition", 2007, http://prtools.org.

[43] C. Eason and K. O'Halloran, "Biomarkers in toxicology versus ecological risk assessment", *Toxicology*, vol. 181, pp. 517–521, 2002.

[44] B. Efron, "Bootstrap methods: another look at the jackknife", *The Annals of Statistics*, vol. 7, no. 1, pp. 1–26, 1979.

[45] J. Ellis, “The superstring–Theory of everything, or of nothing?” *Nature*, vol. 323, pp. 595–598, 1986.

[46] D. Erdogmus and J. Principe, “An error–entropy minimization algorithm for supervised training of nonlinear adaptive systems”, *IEEE Transactions on Signal Processing*, vol. 50, no. 7, pp. 1780–1786, 2002.

[47] D. Erdogmus and J. Principe, “Generalized information potential criterion for adaptive system training”, *IEEE Transactions on Neural Networks*, vol. 13, no. 5, pp. 1035–1044, 2002.

[48] R. Fisher, “The use of multiple measurements in taxonomic problems”, *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[49] V. Forbes, A. Palmqvist, and L. Bach, “The use and misuse of biomarkers in ecotoxicology”, *Environmental Toxicology and Chemistry*, vol. 25, no. 1, pp. 272–280, 2006.

[50] D. Francois, V. Wertz, and M. Verleysen, “Non–Euclidean metrics for similarity search in noisy datasets”, in *Proceedings of the European Symposium on Artificial Neural Networks*. d–side publications, 2005, pp. 339–334.

[51] A. Fraser and D. Burnell, *Computer models in genetics*. New York, USA: McGraw–Hill, 1970.

[52] Y. Freund and R. Schapire, “A desicion–theoretic generalization of on–line learning and an application to boosting”, *Journal of Computer and System Sciences*, vol. 55, pp. 119–139, 1997.

[53] K. Fukunaga and R. Hayes, “The reduced Parzen classifier”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 4, pp. 423–425, 1989.

[54] B. Gabrys, “Neuro–fuzzy approach to processing inputs with missing values in pattern recognition problems”, *International Journal of Approximate Reasoning*, vol. 30, no. 3, pp. 149–179, 2002.

[55] B. Gabrys and L. Petrakieva, “Combining labelled and unlabelled data in the design of pattern classification systems”, *International Journal of Approximate Reasoning*, vol. 35, no. 3, pp. 251–273, 2004.

[56] P. Geladi and B. Kowalski, “Partial least–squares regression: a tutorial”, *Analytica Chimica Acta*, vol. 185, pp. 1–17, 1986.

[57] S. Geman, E. Bienenstock, and R. Doursat, “Neural networks and the bias/variance dilemma”, *Neural computation*, vol. 4, no. 1, pp. 1–58, 1992.

[58] Z. Ghahramani, M. Jordan, J. Cowan, G. Tesauro, and J. Alspector, “Supervised learning from incomplete data via an EM approach”, *Advances in Neural Information Processing Systems*, vol. 6, pp. 120–127, 1994.

[59] E. Gokcay and J. Principe, “Information theoretic clustering”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 158–171, 2002.

[60] E. Goldberg and K. Bertine, “Beyond the Mussel Watch – new directions for monitoring marine pollution”, *The Science of the Total Environment*, vol. 247, no. 2-3, pp. 165–174, 2000.

[61] J. Goldberger, S. Gordon, and H. Greenspan, “An efficient image similarity measure based on approximations of KL–divergence between two Gaussian mixtures”, in *Proceedings of the 9th IEEE International Conference on Computer Vision*, vol. 1. IEEE, 2003, pp. 487–493.

[62] S. Goldman and Y. Zhou, “Enhancing supervised learning with unlabeled data”, in *Proceedings of the 17th International Conference on Machine Learning*. Morgan Kaufmann, 2000, pp. 327–334.

[63] J. Graham, P. Cumsille, and E. Elek-Fisk, “Methods for handling missing data”, *Handbook of psychology*, vol. 2, pp. 87–114, 2003.

[64] H. Hakkoymaz, G. Chatzimilioudis, D. Gunopulos, and H. Mannila, “Applying Electromagnetic Field Theory Concepts to Clustering with Constraints”, in *Machine Learning and Knowledge Discovery in Databases*, ser. Lecture Notes in Computer Science, W. Buntime, M. Grobelnik, D. Mladenic, and J. Shawe-Taylor, Eds. Springer, 2009, vol. 5781, pp. 485–500.

[65] D. Halliday, R. Resnick, and J. Walker, *Fundamentals of physics 8th ed.* New York, USA: John Wiley & Sons, 2007.

[66] D. Hamilton and S. Schladow, “Prediction of water quality in lakes and reservoirs. Part I–model description”, *Ecological Modelling*, vol. 96, no. 1-3, pp. 91–110, 1997.

[67] J. Handl, J. Knowles, and M. Dorigo, “On the performance of ant–based clustering”, in *Design and application of hybrid intelligent systems*, ser. Frontiers in Artificial Intelligence and Applications, A. Abraham, M. Kppen, and K. Franke, Eds. Amsterdam, The Netherlands: IOS Press, 2003, vol. 104, pp. 204–213.

[68] T. Hastie and R. Tibshirani, "Classification by pairwise coupling", *Annals of Statistics*, vol. 26, no. 2, pp. 451–471, 1998.

[69] S. Haykin, *Neural networks: a comprehensive foundation 2nd ed.* New Jersey, USA: Prentice Hall, 1998.

[70] T. Hermann, P. Meinicke, and H. Ritter, "Principal curve sonification", in *Proceeding of the International Conference on Auditory Display*. International Community for Auditory Display, 2000, pp. 81–86.

[71] T. Hermann and H. Ritter, "Listen to your data: Model–based sonification for data analysis", *Advances in intelligent computing and multimedia systems*, pp. 189–194, 1999.

[72] J. Hershey and P. Olsen, "Approximating the Kullback–Leibler divergence between Gaussian mixture models", in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 4. IEEE, 2007, pp. 317–320.

[73] K. Hild, D. Erdogmus, and J. Principe, "Blind source separation using Renyi's mutual information", *IEEE Signal Processing Letters*, vol. 8, no. 6, pp. 174–176, 2003.

[74] K. Hild, D. Pinto, D. Erdogmus, and J. Principe, "Convolutive blind source separation by minimizing mutual information between segments of signals", *IEEE Transactions on circuits and systems–I: Regular papers*, vol. 52, no. 10, pp. 2188–2196, 2005.

[75] S. Hochreiter and M. Mozer, "An Electric Field Approach to Independent Component Analysis", in *Proceedings of the International Workshop on Independent Component Analysis and Blind Signal Separation*, 2000, pp. 45–50.

[76] S. Hochreiter and M. Mozer, "Coulomb classifiers: Reinterpreting SVMs as electrostatic systems", CU-CS-921-01, Department of Computer Science, University of Colorado, Boulder, Tech. Rep., 2001.

[77] S. Hochreiter, M. Mozer, and K. Obermayer, "Coulomb classifiers: Generalizing support vector machines via an analogy to electrostatic systems", *Advances in Neural Information Processing Systems*, vol. 15, pp. 545–552, 2003.

[78] T. Höfer, I. Gerner, U. Gundert-Remy, M. Liebsch, A. Schulte, H. Spielmann, R. Vogel, and K. Wettig, "Animal testing and alternative approaches for the human health risk assessment under the proposed new European chemicals regulation", *Archives of toxicology*, vol. 78, no. 10, pp. 549–564, 2004.

[79] B. Humpert and P. Holley, "Expert systems in finance planning", *Expert Systems*, vol. 5, no. 2, pp. 78–101, 2007.

[80] A. Hunt and T. Hermann, "The importance of interaction in sonification", in *Proceedings of the International Conference on Auditory Display*, 2004.

[81] A. Jain, R. Duin, and J. Mao, "Statistical pattern recognition: A review", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4–37, 2000.

[82] H. Jeffreys, "An invariant form for the prior probability in estimation problems", *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, pp. 453–461, 1946.

[83] R. Jenssen, D. Erdogmus, K. Hild, J. Principe, and T. Eltoft, "Optimizing the Cauchy–Schwarz PDF distance for information theoretic, non–parametric clustering", in *Energy Minimization Methods in Computer Vision and Pattern Recognition*, ser. Lecture Notes in Computer Science, A. Rangarajan, B. Vemurl, and A. Yuille, Eds. Springer, 2005, vol. 3257, pp. 34–45.

[84] R. Jenssen, D. Erdogmus, J. Principe, and T. Eltoft, "The Laplacian spectral classifier", in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2005, pp. 325–328.

[85] R. Jenssen, J. Principe, D. Erdogmus, and T. Eltoft, "The Cauchy–Schwarz divergence and Parzen windowing: Connections to graph theory and Mercer kernels", *Journal of the Franklin Institute*, vol. 343, no. 6, pp. 614–629, 2006.

[86] K. Jeong and J. Principe, "Enhancing the correntropy MACE filter with random projections", *Neurocomputing*, vol. 72, no. 1-3, pp. 102–111, 2008.

[87] I. Jolliffe, *Principal component analysis 2nd ed.* New York, USA: Springer Verlag, 2002.

[88] M. C. Jones, J. S. Marron, and S. 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. [Online]. Available: http://www.jstor.org/stable/2291420

[89] J. Kapur, *Measures of information and their applications*. New York, USA: John Wiley & Sons, 1994.

[90] S. Kirkpatrick, "Optimization by simulated annealing: Quantitative studies", *Journal of Statistical Physics*, vol. 34, no. 5, pp. 975–986, 1984.

[91] G. Klopman, H. Zhu, M. Fuller, and R. Saiakhov, "Searching for an enhanced predictive tool for mutagenicity", *SAR and QSAR in Environmental Research*, vol. 15, no. 4, pp. 251–263, 2004.

[92] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection", in *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, vol. 2, no. 12. Morgan Kaufmann, 1995, pp. 1137–1145.

[93] K. Korb and A. Nicholson, *Bayesian artificial intelligence*. Chapman & Hall, 2004.

[94] R. Kothari and V. Jain, "Learning from labeled and unlabeled data using a minimal number of queries", *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1496–1505, 2003.

[95] S. Kullback, *Information theory and statistics*. New York, USA: Dover Publications Inc., 1997.

[96] S. Kullback and R. Leibler, "On information and sufficiency", *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[97] L. Kuncheva, *Fuzzy classifier design*. Heidelberg, Germany: Physica Verlag, 2000.

[98] L. Kuncheva, *Combining pattern classifiers: methods and algorithms*. New York, USA: John Wiley & Sons, 2004.

[99] L. Kuncheva and C. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy", *Machine Learning*, vol. 51, no. 2, pp. 181–207, 2003.

[100] L. Le Cam and G. Yang, *Asymptotics in statistics: some basic concepts*. New York, USA: Springer Verlag, 2000.

[101] J. Lin, "Divergence measures based on the Shannon entropy", *IEEE Transactions on Information theory*, vol. 37, no. 1, pp. 145–151, 1991.

[102] W. Liu, P. Pokharel, and J. Principe, "Correntropy: A Localized Similarity Measure", in *Proceedings of the International Joint Conference on Neural Networks*, 2006, pp. 4919–4924.

[103] W. Liu, P. Pokharel, and J. Principe, "Error Entropy, Correntropy and M–Estimation", in *Proceedings of the 16th IEEE Signal Processing Society Workshop on Machine Learning for Signal Processing*. IEEE, 2006, pp. 179–184.

[104] W. Liu, P. Pokharel, and J. Principe, "Correntropy: properties and applications in non–Gaussian signal processing", *IEEE Transactions on Signal Processing*, vol. 55, no. 11, pp. 5286–5298, 2007.

[105] D. Livingstone, J. Chipman, D. Lowe, C. Minier, and R. Pipe, "Development of biomarkers to detect the effects of organic pollution on aquatic invertebrates: recent molecular, genotoxic, cellular and immunological studies on the common mussel (Mytilus edulis L.) and other mytilids", *International Journal of Environment and Pollution*, vol. 13, no. 1, pp. 56–91, 2000.

[106] D. MacKay, *Information theory, inference, and learning algorithms*. Cambridge, UK: Cambridge University Press, 2003.

[107] W. Madow, *Incomplete data in sample surveys*. Academic Press, 1983.

[108] H. Maier and G. Dandy, "The use of artificial neural networks for the prediction of water quality parameters", *Water Resources Research*, vol. 32, no. 4, pp. 1013–1022, 1996.

[109] T. Minka, "A family of algorithms for approximate Bayesian inference", Ph.D. dissertation, MIT, 2001.

[110] T. Mitchell, *Machine learning*. USA: McGraw Hill, 1997.

[111] T. Mitchell, "The role of unlabeled data in supervised learning", in *Proceedings of the 6th International Colloquium on Cognitive Science*, 1999.

[112] P. Moreno, P. Ho, and N. Vasconcelos, "A Kullback–Leibler divergence based kernel for SVM classification in multimedia applications", *Advances in Neural Information Processing Systems*, vol. 16, pp. 1385–1392, 2004.

[113] T. Netzeva and T. Schultz, "QSARs for the aquatic toxicity of aromatic aldehydes from Tetrahymena data", *Chemosphere*, vol. 61, no. 11, pp. 1632–1643, 2005.

[114] K. Nigam and R. Ghani, "Understanding the behavior of co–training", in *Proceedings of the KDD–2000 Workshop on Text Mining*, 2000.

[115] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions", *Pattern Recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[116] W. Ott, A. Steinemann, and L. Wallace, *Exposure analysis*. Boca Raton, USA: CRC Press, 2006.
[117] W. Outhwaite and S. Turner, *Handbook of Social Science Methodology*. London, UK: SAGE Publications Ltd., 2007.
[118] M. Pace, "Prediction and the aquatic sciences", *Canadian Journal of Fisheries and Aquatic Sciences*, vol. 58, no. 1, pp. 63–72, 2001.
[119] D. Pampanin, E. Ravagnan, S. Apeland, N. Aarab, B. Godal, S. Westerlund, D. Hjermann, T. Eftestøl, M. Budka, B. Gabrys, A. Viarengo, and J. Barsiene, "The Marine Environment I.Q. concept. Developing an Index of the Quality of the Marine Environment based on biomarkers: integration of pollutant effects on marine organisms." in *Proceedings of the 27th ESCPB (New European Society for Comparative Physiology and Biochemistry) Congress*, 2010 (Accepted).
[120] L. Paninski, "Estimation of entropy and mutual information", *Neural Computation*, vol. 15, no. 6, pp. 1191–1253, 2003.
[121] E. Parzen, "On estimation of a probability density function and mode", *The annals of mathematical statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.
[122] D. Peakall, "The role of biomarkers in environmental assessment (1). Introduction", *Ecotoxicology*, vol. 3, no. 3, pp. 157–160, 1994.
[123] W. Pedrycz and J. Waletzky, "Fuzzy clustering with partial supervision", *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 27, no. 5, pp. 787–795, 1997.
[124] F. Perez–Cruz, "Kullback–Leibler divergence estimation of continuous distributions", in *Proceedings of the IEEE International Symposium on Information Theory*. IEEE, 2008, pp. 1666–1670.
[125] J. Principe, *Information Theoretic Learning: Renyi's Entropy and Kernel Perspectives*. New York, USA: Springer, 2010.
[126] J. Principe, N. Euliano, and W. Lefebvre, *Neural and adaptive systems: Fundamentals through simulations*. New York, USA: John Wiley & Sons, 1999.
[127] J. Principe, D. Xu, and J. Fisher, "Information Theoretic Learning", in *Unsupervised Adaptive Filtering*, S. Haykin, Ed. John Wiley & Sons, 2000, pp. 265–319.
[128] J. Principe, D. Xu, Q. Zhao, and J. Fisher, "Learning from Examples with Information Theoretic Criteria", *The Journal of VLSI Signal Processing*, vol. 26, no. 1, pp. 61–77, 2000.
[129] S. Rao, W. Liu, J. Principe, and A. de Medeiros Martins, "Information theoretic mean shift algorithm", in *Proceedings of the 16th IEEE Signal Processing Society Workshop on Machine Learning for Signal Processing*. IEEE, 2006, pp. 155–160.
[130] V. C. Raykar and R. Duraiswami, "Fast optimal bandwidth selection for kernel density estimation", in *Proceedings of the 6th SIAM International Conference on Data Mining*, J. Ghosh, D. Lambert, D. Skillicorn, and J. Srivastava, Eds., 2006, pp. 524–528.
[131] K. Reckhow, "Water quality prediction and probability network models", *Canadian Journal of Fisheries and Aquatic Sciences*, vol. 56, no. 7, pp. 1150–1158, 1999.
[132] A. Richard, "Future of toxicology–predictive toxicology: An expanded view of chemical toxicity", *Chemical Research in Toxicology*, vol. 19, no. 10, pp. 1257–1262, 2006.
[133] A. Richard and R. Benigni, "AI and SAR approaches for predicting chemical carcinogenicity: survey and status report", *SAR and QSAR in Environmental Research*, vol. 13, no. 1, pp. 1–19, 2002.
[134] B. Ripley, *Pattern recognition and neural networks*. Cambridge, UK: Cambridge University Press, 1996.
[135] N. Roy and A. McCallum, "Toward optimal active learning through sampling estimation of error reduction", in *Proceedings of the 18th International Conference on Machine Learning*. Mo, 2001, pp. 441–448.
[136] D. Rubin, "Inference and missing data", *Biometrika*, vol. 63, no. 3, pp. 581–592, 1976.
[137] D. Rubin, *Multiple Imputation for Nonresponse in Surveys*. New York, USA: John Wiley & Sons, 1987.
[138] D. Ruta, "Classifier diversity in combined pattern recognition systems", Ph.D. dissertation, University of Paisley, 2003.
[139] D. Ruta and B. Gabrys, "An overview of classifier fusion methods", *Computing and Information Systems*, vol. 7, no. 1, pp. 1–10, 2000.

[140] D. Ruta and B. Gabrys, "Analysis of the correlation between majority voting error and the diversity measures in multiple classifier systems", in *Proceedings of the 4th International Symposium on Soft Computing*, 2001, pp. 1824–025.

[141] D. Ruta and B. Gabrys, "The Boundaries of Knowledge", *Computing and Information Systems*, vol. 8, no. 3b, pp. 109–110, 2001.

[142] D. Ruta and B. Gabrys, "A theoretical analysis of the limits of majority voting errors for multiple classifier systems", *Pattern Analysis & Applications*, vol. 5, no. 4, pp. 333–350, 2002.

[143] D. Ruta and B. Gabrys, "Physical field models for pattern classification", *Soft Computing*, vol. 8, no. 2, pp. 126–141, 2003.

[144] D. Ruta and B. Gabrys, "Nature Inspired Learning Models", in *Proceedings of the European Symposium on Nature Inspired Smart Information Systems*, 2005.

[145] D. Ruta and B. Gabrys, "Neural network ensembles for time series prediction", in *Proceedings of the International Joint Conference on Neural Networks*. IEEE, 2007, pp. 1204–1209.

[146] D. Ruta and B. Gabrys, "Reducing spatial data complexity for classification models", *Computational Methods in Science and Engineering*, vol. 963, pp. 603–613, 2007.

[147] D. Ruta and B. Gabrys, "A framework for machine learning based on dynamic physical fields", *Natural Computing*, vol. 8, no. 2, pp. 219–237, 2009.

[148] I. Santamaría, P. Pokharel, and J. Principe, "Generalized correlation function: Definition, properties, and application to blind equalization", *IEEE Transactions on Signal Processing*, vol. 54, no. 6, pp. 2187–2197, 2006.

[149] W. Sarle, "Prediction with missing inputs", in *Proceedings of the 4th Joint Conference on Information Sciences*, vol. 98, 1998, pp. 399–402.

[150] J. Schafer and J. Graham, "Missing data: Our view of the state of the art", *Psychological Methods*, vol. 7, no. 2, pp. 147–177, 2002.

[151] J. Schafer and N. Schenker, "Inference with imputed conditional means", *Journal of the American Statistical Association*, vol. 95, no. 449, pp. 144–154, 2000.

[152] R. Schapire, Y. Freund, P. Bartlett, and W. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods", *The annals of statistics*, vol. 26, no. 5, pp. 1651–1686, 1998.

[153] T. Schultz and T. Netzeva, "Development and evaluation of QSARs for ecotoxic endpoints: The benzene response–surface model for Tetrahymena toxicity", *Predicting Chemical Toxicity and Fate*, pp. 265–284, 2004.

[154] S. Seth and J. Principe, "Compressed signal reconstruction using the correntropy induced metric", in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2008, pp. 3845–3848.

[155] C. Shannon, "A mathematical theory of communication", *Bell System Technical Journal*, vol. 27, no. July, October, pp. 379–423, 623–656, 1948.

[156] A. Sharkey, *Combining Artificial Neural Nets: Ensemble and Modular Multi–net Systems*. New York, USA: Springer–Verlag, 1999.

[157] S. J. Sheather and M. C. Jones, "A reliable data–based bandwidth selection method for kernel density estimation", *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 53, no. 3, pp. 683–690, 1991. [Online]. Available: http://www.jstor.org/stable/2345597

[158] B. Silverman, *Density estimation for statistics and data analysis*. Boca Raton, USA: Chapman & Hall/CRC Press, 1998.

[159] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos, "Online outlier detection in sensor data using non–parametric models", in *Proceedings of the 32nd international conference on Very large data bases*. VLDB Endowment, 2006, pp. 187–198.

[160] I. Tetko, G. Poda, C. Ostermann, and R. Mannhold, "Accurate In Silico log P Predictions: One Can't Embrace the Unembraceable", *QSAR & Combinatorial Science*, vol. 28, no. 8, pp. 845–849, 2009.

[161] I. Tetko, I. Sushko, A. Pandey, H. Zhu, A. Tropsha, E. Papa, T. O berg, R. Todeschini, D. Fourches, and A. Varnek, "Critical assessment of QSAR models of environmental toxicity against tetrahymena pyriformis: focusing on applicability domain and overfitting by variable selection", *Journal of Chemical Information and Modeling*, vol. 48, no. 9, pp. 1733–1746, 2008.

[162] S. Theodoridis and K. Koutroumbas, *Pattern Recognition 4th ed.* San Diego, USA: Academic Press, 2009.

[163] R. Tibshirani, "Regression shrinkage and selection via the lasso", *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[164] A. Tikhonov and V. Arsenin, *Solutions of ill–posed problems*. Washington, USA: VH Winston, 1977.

[165] K. Torkkola, "Visualizing class structure in data using mutual information", in *Proceeding of the IEEE Signal Processing Society Workshop*, 2000, pp. 376–385.

[166] K. Torkkola, "Nonlinear feature transforms using maximum mutual information", in *Proceedings of the International Joint Conference on Neural Networks*, vol. 4, 2001, pp. 2756–2761.

[167] K. Torkkola, "Learning discriminative feature transforms to low dimensions in low dimensions", in *Advances in Neural Information Processing Systems 14*. Cambridge, USA: MIT Press, 2002.

[168] K. Torkkola, "On feature extraction by mutual information maximization", in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2002.

[169] K. Torkkola, "Feature extraction by non parametric mutual information maximization", *Journal of Machine Learning Research*, vol. 3, pp. 1415–1438, 2003.

[170] K. Torkkola and W. Campbell, "Mutual information in learning feature transformations", in *Proceedings of the 17th International Conference on Machine Learning*. Morgan Kaufmann, 2000, pp. 1015–1022.

[171] V. Tresp, S. Ahmad, and R. Neuneier, "Training neural networks with deficient data", *Advances in Neural Information Processing Systems*, vol. 6, pp. 128–135, 1994.

[172] A. Tsymbal, "The problem of concept drift: definitions and related work", *Informe técnico: TCD-CS-2004-15, Departament of Computer Science Trinity College, Dublin, http://www.scss.tcd.ie/publications/tech-reports/reports.04*, vol. 4, pp. 9–15, 2004.

[173] S. Tulyakov, S. Jaeger, V. Govindaraju, and D. Doermann, "Review of Classifier Combination Methods", in *Machine Learning in Document Analysis and Recognition*, ser. Studies in Computational Intelligence, S. Marinai and H. Fujisawa, Eds. Springer, 2008, pp. 361–386.

[174] B. Turlach, "Bandwidth selection in kernel density estimation: A review", *CORE and Institut de Statistique*, pp. 23–493, 1993.

[175] F. Van der Heijden, R. Duin, D. De Ridder, and D. Tax, *Classification, parameter estimation, and state estimation: an engineering approach using MATLAB*. New York, USA: John Wiley & Sons, 2004.

[176] V. Vapnik, *Statistical learning theory*. New York, USA: John Wiley & Sons, 1998.

[177] V. Vapnik, *The nature of statistical learning theory*. New York, USA: Springer Verlag, 2000.

[178] S. Verboven and M. Hubert, "LIBRA: a MATLAB library for robust analysis", *Chemometrics and Intelligent Laboratory Systems*, vol. 75, no. 2, pp. 127–136, 2005.

[179] Q. Wang, S. Kulkarni, and S. Verdu, "A nearest–neighbor approach to estimating divergence between continuous random vectors", in *Proceedings of the IEEE International Symposium on Information Theory*. IEEE, 2006, pp. 242–246.

[180] A. Webb, *Statistical pattern recognition*. New York, USA: John Wiley & Sons, 2002.

[181] S. Weiss and C. Kulikowski, *Computer systems that learn: classification and prediction methods from statistics, neural nets, machine learning, and expert systems*. San Francisco, USA: Morgan Kaufmann, 1991.

[182] S. Wold, M. Sjöström, and L. Eriksson, "PLS–regression: a basic tool of chemometrics", *Chemometrics and intelligent laboratory systems*, vol. 58, no. 2, pp. 109–130, 2001.

[183] L. Zadeh, "Toward a generalized theory of uncertainty (GTU)–an outline", *Information Sciences*, vol. 172, no. 1-2, pp. 1–40, 2005.

[184] S. Zhou and R. Chellappa, "Kullback–Leibler distance between two Gaussian densities in reproducing kernel Hilbert space", in *Proceedings of the IEEE International Symposium on Information Theory*. IEEE, 2004.

[185] H. Zhu, A. Tropsha, D. Fourches, A. Varnek, E. Papa, P. Gramatica, T. O berg, P. Dao, A. Cherkasov, and I. Tetko, "Combinatorial QSAR modeling of chemical toxicants tested against Tetrahymena pyriformis", *Journal of Chemical Information and Modeling*, vol. 48, no. 4, pp. 766–784, 2008.

[186] W. Zurek, *Complexity, Entropy and the Physics of Information*. Redwood City, USA: Westview Press, 1989.