Department of Information and Computer Science

Machine learning methods for incomplete data and variable selection

Emil Eirola

Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 144/2014

Machine learning methods for incomplete data and variable selection

Emil Eirola

Doctoral dissertation for the degree of Doctor of Science in Techology to be presented with due permission of the School of Science for public examination and debate in Auditorium T2 at the Aalto University School of Science (Espoo, Finland) on the 17th of October 2014 at 12 noon.

Aalto University
School of Science
Department of Information and Computer Science
Environmental and Industrial Machine Learning Group

Supervising professor
Prof. Juha Karhunen

Thesis advisor
Dr. Amaury Lendasse

Preliminary examiners
Assoc. Prof. Alberto Guillén, University of Granada, Spain
Prof. Dr. Barbara Hammer, Bielefeld University, Germany

Opponent
Prof. Fabrice Rossi, University Paris 1 Panthéon-Sorbonne, France

Aalto University publication series
DOCTORAL DISSERTATIONS 144/2014

ISBN 978-952-60-5870-2
ISBN 978-952-60-5871-9 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-60-5871-9

Images: ESA/Herschel/PACS/MESS Key Programme Supernova Remnant Team; NASA, ESA and Allison Loll/Jeff Hester (Arizona State University)

Unigrafia Oy
Helsinki 2014

Finland

441 697
Printed matter

A! **Aalto University**

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Abstract

Author
Emil Eirola

Name of the doctoral dissertation
Machine learning methods for incomplete data and variable selection

Publisher School of Science

Unit Department of Information and Computer Science

Series Aalto University publication series DOCTORAL DISSERTATIONS 144/2014

Field of research Computer and Information Science

Manuscript submitted 10 June 2014 **Date of the defence** 17 October 2014

Permission to publish granted (date) 14 August 2014 **Language** English

☐ **Monograph** ☒ **Article dissertation (summary + original articles)**

Abstract

Machine learning is a rapidly advancing field. While increasingly sophisticated statistical methods are being developed, their use for concrete applications is not necessarily clear-cut. This thesis explores techniques to handle some issues which arise when applying machine learning algorithms to practical data sets. The focus is on two particular problems: how to effectively make use of incomplete data sets without having to discard samples with missing values, and how to select an appropriately representative set of variables for a given task.

For tasks with missing values, distance estimation is presented as a new approach which would directly enable a large class of machine learning methods to be used. It is shown that the distance can be estimated reliably and efficiently, and experimental results are provided to support the procedure. The idea is studied both on a general level, as well as how to conduct the estimation with a Gaussian mixture model.

The issue of variable selection is considered from the perspective of finding suitable criteria which are feasible to calculate and effective at distinguishing the most useful variables also for non-linear connections when limited data is available. Two alternatives are studied, the first being the Delta test, which is a noise variance estimator based on the nearest neighbour regression model. It is shown that the optimal selection of feature uniquely minimises the expectation of the estimator. The second method is a mutual information estimator based on a mixture of Gaussians. The procedure is based on a single mixture model which can be used to derive estimates for any subset of variables. This leads to congruous estimates for the mutual information of different variable sets, which can then be compared to each other in a meaningful way to find the optimal.

The Gaussian mixture model proves to be a highly useful tool for several tasks, especially concerning data with missing values. In this thesis, it is used for distance estimation, time series modelling, and mutual information estimation for variable selection.

Keywords Machine learning, missing values, variable selection, Gaussian mixture model, mutual information, Delta test

ISBN (printed) 978-952-60-5870-2 **ISBN (pdf)** 978-952-60-5871-9

ISSN-L 1799-4934 **ISSN (printed)** 1799-4934 **ISSN (pdf)** 1799-4942

Location of publisher Helsinki **Location of printing** Helsinki **Year** 2014

Pages 167 **urn** http://urn.fi/URN:ISBN:978-952-60-5871-9

Aalto-universitetet

Sammandrag

Aalto-universitetet, PB 11000, 00076 Aalto www.aalto.fi

Författare
Emil Eirola

Doktorsavhandlingens titel
Metoder för ofullständiga datamängder och attributval i maskininlärning

Utgivare Högskolan för teknikvetenskaper

Enhet Institutionen för datavetenskap

Seriens namn Aalto University publication series DOCTORAL DISSERTATIONS 144/2014

Forskningsområde Informationsteknik

Inlämningsdatum för manuskript 10.06.2014 **Datum för disputation** 17.10.2014

Beviljande av publiceringstillstånd (datum) 14.08.2014 **Språk** Engelska

☐ **Monografi** ☒ **Sammanläggningsavhandling (sammandrag plus separata artiklar)**

Sammandrag

Maskininlärning är ett snabbt framåtgående forskningsområde. Samtidigt som allt mer avancerade statistiska metoder kommer fram, är deras tillämpning för konkreta användningsområden inte nödvändigtvis entydigt. Denna avhandling utforskar metoder för att hantera vissa problem som uppkommer vid tillämpning av maskininlärningsalgoritmer till praktiska datamängder. Tyngdpunkten ligger på två särskilda svårigheter: hur att effektivt utnyttja ofullständiga datamängder utan att lämna bort de datapunkter som saknar värden, och hur att välja en representativ grupp av attribut (variabler) för ett visst modelleringsuppdrag.

För uppgifter med saknade värden, presenteras avståndsestimering som en ny strategi som direkt skulle möjliggöra användningen en stor mängd maskininlärningsmetoder. Det visar sig att det går att uppskatta avstånden tillförlitligt och kostnadeffektivt. Idén behandlas både på en allmän nivå och hur man utför estimeringen med en Gaussisk blandningsmodell.

Frågan om attributval beaktas utgående från passliga kriterier som är lämpliga att beräkna och effektiva på att identifiera de mest användbara variablerna även för ickelinjära modeller och när den tillgängliga datamängden är begränsad. Två alternativ undersöks: den första är Delta-testet, baserad på den närmaste grannens regressionsanalys. Det visas att det optimala valet av variabler minimerar Delta-testets väntevärde. Den andra metoden är en estimator av ömsesidig information baserad på den Gaussiska blandningsmodellen. Tekniken använder sig av en enda blandningsmodell, som kan tillämpas för att härleda uppskattningar för diverse urval av variabler. Detta leder till motsvarande beräkningar av den ömsesidiga informationen för olika variabeluppsättningar, som sedan kan jämföras med varandra för att hitta den optimala.

Den Gaussiska blandningsmodellen visar sig vara ett högt användbart redskap för flera tillfällen, särskilt angående data som saknar värden. I denna avhandling används den för avståndsestimering, modellering av tidsserier, och estimering av ömsesidig information för attributval.

Nyckelord Maskininlärning, neuronnät, saknade värden, attributval, Gaussisk blandfördelning, ömsesidig information, Delta-testet

ISBN (tryckt) 978-952-60-5870-2 **ISBN (pdf)** 978-952-60-5871-9

ISSN-L 1799-4934 **ISSN (tryckt)** 1799-4934 **ISSN (pdf)** 1799-4942

Utgivningsort Helsingfors **Tryckort** Helsingfors **År** 2014

Sidantal 167 **urn** http://urn.fi/URN:ISBN:978-952-60-5871-9

Preface

The work for this thesis has been conducted at the Department of Information and Computer Science at the Aalto University School of Science during the years 2009–2014. I am very grateful to both my current supervisor Juha Karhunen and original supervisor Olli Simula for the opportunity and privilege to be allowed to work on the thesis on my own terms until completion.

I want to thank my instructor Amaury "Momo" Lendasse for his dedication to his students, and the guidance and practical wisdom he has provided. Momo originally encouraged me to pursue a doctoral degree and has constantly made sure I am on track for completing it even when plans and research directions have changed.

I thank the pre-examiners Barbara Hammer and Alberto Guillén for their supportive comments and valuable insight which helped improve the final draft of this thesis. For my thesis defence, I am very honoured to have Fabrice Rossi as the opponent.

This work would not have been possible without all my colleagues and friends in the Environmental and Industrial Machine Learning group. I am particularly indebted to Yoan Miche, Francesco Corona, Dušan Sovilj, Mark van Heeswijk, Antti Sorjamaa, Elia Liitiäinen, Yu Qi, Federico Montesino Pouzols, Laura Kainulainen, Alexander Grigorievskiy, and Luiza Sayfullina. Thank you all!

I am proud and grateful to have been part of the Finnish Doctoral Programme in Computational Sciences (FICS), which has been the primary source of funding for my work. I want to thank FICS coordinator Ella Bingham for the help. Personal grants from the Nokia Foundation and the Emil Aaltonen Foundation have also been appreciated contributions to my research.

Most of all, I am deeply grateful to my mom and dad, my brothers Axel and Oskar, and all my friends for all the support they have shown over the years.

Helsinki, September 16, 2014,

Emil Eirola

Contents

List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.

I Emil Eirola, Gauthier Doquire, Michel Verleysen, and Amaury Lendasse. Distance Estimation in Numerical Data Sets with Missing Values. *Information Sciences*, volume 240, pages 115–128, 2013.

II Emil Eirola, Amaury Lendasse, Vincent Vandewalle and Christophe Biernacki. Mixture of Gaussians for distance estimation with missing data. *Neurocomputing*, volume 131, pages 32–42, 2014.

III Qi Yu, Yoan Miche, Emil Eirola, Mark van Heeswijk, Eric Séverin, and Amaury Lendasse. Regularized Extreme Learning Machine For Regression with Missing Data. *Neurocomputing*, volume 102, pages 45–51, 2013.

IV Emil Eirola and Amaury Lendasse. Gaussian Mixture Models for Time Series Modelling, Forecasting, and Interpolation. In *Advances in Intelligent Data Analysis XII – 12th International Symposium (IDA 2013)*, LNCS volume 8207, pages 162–173, October 2013.

V Emil Eirola, Amaury Lendasse, and Juha Karhunen. Variable Selection for Regression Problems Using Gaussian Mixture Models to Estimate Mutual Information. In *The 2014 International Joint Conference on Neural Networks (IJCNN 2014)*, pages 1606–1613, July 2014.

VI Emil Eirola, Elia Liitiäinen, Amaury Lendasse, Francesco Corona, and Michel Verleysen. Using the Delta test for variable selection. In *European Symposium on Artificial Neural Networks (ESANN 2008)*, pages 25–30, April 2008.

VII Emil Eirola, Amaury Lendasse, Francesco Corona, and Michel Verleysen. The Delta Test: The 1-NN Estimator as a Feature Selection Criterion. In *The 2014 International Joint Conference on Neural Networks (IJCNN 2014)*, pages 4214–4222, July 2014.

List of Abbreviations

AIC	Akaike Information Criterion
BIC	Bayesian Information Criterion
DT	The Delta Test
ELM	Extreme Learning Machine
EM	The Expectation–Maximisation algorithm
ESD	Expected Squared Distance
GMM	Gaussian Mixture Model
HDDC	High-Dimensional Data Clustering
ICkNNI	Incomplete-case k-NN imputation
LARS	Least Angle Regression
LOO	Leave-one-out (cross-validation)
LS-SVM	Least Squares Support Vector Machines
MAR	Missing at Random
MCAR	Missing Completely at Random
MDS	Multi-Dimensional Scaling
MI	Mutual Information
ML	Maximum Likelihood
MLMI	Maximum Likelihood Mutual Information
MSE	Mean Squared Error
NN	Nearest Neighbour
PDS	Partial Distance Strategy
RBF	Radial Basis Function
RMSE	Root Mean Squared Error
SVM	Support Vector Machines

List of Notations

$\boldsymbol{x}$, $\boldsymbol{y}$	Vectors
$\mathbf{X}$, $\mathbf{Y}$	Matrices
$\boldsymbol{x}^T$	Transpose of $\boldsymbol{x}$
$x_{i,l}$	Element l of an indexed vector $\boldsymbol{x}_i$
X, Y	Random variables
$\lvert\cdot\rvert$	Absolute value; cardinality (of a set)
$\lVert\cdot\rVert$	The Euclidean (L^2) norm
$\lVert\cdot\rVert_1$	The L^1 norm
$\lVert\cdot\rVert_F$	The Frobenius norm of a matrix
$\mathbf{0}$	Matrix or vector where each element is 0
$\mathbf{1}$	Matrix or vector where each element is 1
$\mathrm{Cov}[\cdot]$	Covariance matrix of a vector-valued random variable
$\mathrm{Cov}[\cdot,\cdot]$	Covariance of two random variables
d	Dimension of the input space
$d(\cdot,\cdot)$	Distance
$\hat{d}(\cdot,\cdot)$	Estimated distance
$\det(\cdot)$	Determinant
$\delta(\cdot)$	Delta test
$\mathrm{E}[\cdot]$	Expectation
ε	Random variable signifying noise
$H(\cdot)$	Entropy
$\mathbf{I}$	Identity matrix
$I(\cdot;\cdot)$	Mutual information
K	Number of components in a Gaussian mixture
$K(\cdot,\cdot)$	Kernel

$\mathscr{L}(\boldsymbol{\theta};\mathbf{X})$	Likelihood function
λ	Eigenvalue
$\boldsymbol{\Lambda}$	Eigenvalue matrix
M_i	Index set of missing values in $\boldsymbol{x}_i$
$\boldsymbol{\mu}_k$	Mean of a component k
N	Number of samples in a data set
$\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$	The multivariate normal density function
NN(i)	Nearest neighbour of $\boldsymbol{x}_i$
O_i	Index set of observed values of $\boldsymbol{x}_i$
Ω	Space of model parameters
$p(\cdot)$	Probability density
P	Number of free parameters in a model
$\mathscr{P}(\cdot)$	Power set
π_k	Mixing coefficient of a component k
Φ	Restricted space of model parameters
$\mathbf{Q}$	Eigenvector matrix
$Q(\theta \mid \theta^{(t)})$	Expected log-likelihood
$\mathbf{R}$	Autocovariance matrix
$\mathbb{R}$	Field of real numbers
$\boldsymbol{\Sigma}_k$	Covariance matrix of a component k
t_{ik}	Probability of sample $\boldsymbol{x}_i$ to be in component k
tr($\cdot$)	Trace of a matrix
θ	Set of parameters
Var[$\cdot$]	Variance
$\boldsymbol{w}$	Weight vector

1. Introduction

1.1 Aims and scope

The prevalence of machine learning has been steadily increasing in the current information age. Engineering advances in processor performance and storage capacities have provided an opportunity to make practical use of computational statistics on a large scale. Simultaneously, the research community has contributed by devising new clever algorithms to maximise the amount of relevant information that can be extracted from data. While data sets are large at times, the more common situation is that the number of samples is limited by practical issues, meaning that all of the available data must be used as efficiently as possible in order to achieve the desired results. This thesis strives to show how some of these difficulties can be addressed in machine learning contexts.

One pertinent issue is incomplete data sets, where some samples have missing information. Most machine learning techniques are not designed to work with such data, and the unfortunate consequence is often that the incomplete data samples are simply ignored in the subsequent analysis. A primary aim of the research for this thesis has been to find better ways to deal with this data, leading to the approach of directly estimating the distances between data samples. The publications show that estimating distances is feasible in general, and a specific method to conduct it with Gaussian mixtures is introduced. The resulting estimates can then be incorporated into one of several machine learning procedures, since many of them can be formulated in terms of the differences or similarities between data points.

Time series with gaps are a common special case of incomplete data, and a frequent occurrence in many fields. In this thesis, the problem

of modelling such data is considered by applying the Gaussian mixture model with appropriate restrictions on the covariance matrices to match the autocovariance structure of a time series.

The increasing size of the data sets leads to the necessity of variable selection. Automated measurement systems can efficiently gather large batches of information about individual targets, but it can be unfeasible to determine which features are relevant for a particular task. Machine learning methods generally consider each input variable to be of equal importance, although this is not necessarily true. Variables may provide redundant information already covered by other variables in a better form, or be entirely irrelevant in certain situations. This thesis presents two criteria to determine which input variables or sets of variables are most useful for a given modelling task. The first is the Delta test, which uses the average error of the nearest neighbour model as a relative measure of quality of the involved variables. A more sophisticated alternative is achieved by estimating the mutual information between input and output variables. While mutual information has been used for variable selection previously, Publication V of this thesis presents a new procedure to calculate it using Gaussian mixtures. This estimation method is particularly suitable for comparing estimates over different variable sets.

The Gaussian mixture model is a recurring topic due to its general usefulness as a probability density estimate, but it is particularly relevant due to how data with missing values can be included when fitting the model.

1.2 Publications and author's contribution

This thesis consists of seven publications written with coauthors. The contributions of the present author to each article are detailed here.

Publication I: Distance Estimation in Numerical Data Sets with Missing Values This first article introduces the idea of distance estimation as a useful new approach to machine learning with missing data. It is shown that calculating the expectation of the squared distance between samples reduces to finding the expectation and variance separately of each missing value, and a procedure to conduct the estimation based on the principle of maximum entropy is presented. Directly estimating the distances leads to more accurate results than using an equivalent model to fill in the miss-

ing values and calculating distances on the imputed data. The author contributed with the original idea, conducted the experiments, and wrote the article.

Publication II: Mixture of Gaussians for distance estimation with missing data This article is about using Gaussian mixtures to more accurately conduct the distance estimation, also covering several issues of fitting the mixture model to data with missing values. In addition, it includes some results on how the estimated distances are used for building extreme learning machine neural networks. The author contributed with the original idea, conducted the experiments, and wrote the manuscript.

Publication III: Regularized Extreme Learning Machine For Regression with Missing Data Using the estimated distances for an extreme learning machine is further explored here with various regularisation techniques. The article also includes results for a financial application, using the model to predict the possibility bankruptcy for a collection of companies. The author was responsible for the idea and programming implementation concerning missing values and distance estimation (25% of the total work).

Publication IV: Gaussian Mixture Models for Time Series Modelling, Forecasting, and Interpolation The article studies how Gaussian mixtures can be used for time series modelling by appropriate constraints on the model parameters. A particular focus is placed on modelling incomplete (i.e., *gapped*) time series. The author contributed with the original idea, conducted the experiments, and wrote the manuscript.

Publication V: Variable Selection for Regression Problems Using Gaussian Mixture Models to Estimate Mutual Information A further use of the Gaussian mixture model is for mutual information estimation. Since a single mixture model can be used to estimate the mutual information for different subsets of variables, this is particularly useful for variable selection, and also works directly on data with missing values. The author contributed with the original idea, conducted the experiments, and wrote the manuscript.

Publication VI: Using the Delta test for variable selection The Delta test is a relatively simple method that had been used occasionally as a variable selection criterion previously, but this is the first article to specifically focus on studying the reasons why it works as well as it does. The author contributed with the original idea, conducted the experiments, and wrote the manuscript with input from the co-authors (80% of the total work).

Publication VII: The Delta Test: The 1-NN Estimator as a Feature Selection Criterion This extended update of Publication VI contains a considerably more thorough theoretical analysis and experimental evaluation of the same method. The author contributed with the original idea, conducted the experiments, and wrote the manuscript with input from the co-authors (80% of the total work).

1.3 Structure of the thesis

The remainder of this thesis includes four chapters. Chapter 2 presents an overview of relevant background information, including concepts in the field of machine learning and current approaches to solving the studied issues of incomplete data and variable selection. Chapter 3 introduces the relevant contributions to machine learning with missing data from the publications. The two methods for variable selection are studied in Chapter 4. Conclusions and summary are contained in Chapter 5.

2. Background

2.1 Overview of machine learning

Machine learning is the study of designing and constructing methods to learn from data. In this context, learning refers to inferring statistical properties and relationships from a collection of data samples [79]. The intent is that the discovered patterns can be used to predict associated values for future data.

The field of machine learning – or *pattern recognition*, as it is also known as – is generally divided into supervised and unsupervised learning. In *supervised learning*, data samples with labels are available for the learning algorithm, and the goal is to find an accurate predictive model in the form of an algorithm which takes the sample as an input and returns the correct label [2]. The training data generally consists of a set of input-output pairs $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, where each input sample $\boldsymbol{x}_i$ is a d-dimensional vector of numbers, and each sample has an associated output label y_i. These labels can take different forms, such as representing membership of a class, a numerical quantity, or a vector of several such properties. The d different values specified for each input are known as *variables* (or *features*, *attributes*, or *covariates*). The main examples of supervised learning tasks are *classification*, where the goal is to assign input points to one of two or more classes, and *regression*, where the target output is a numerical quantity, often a continuous variable.

The other main category, *unsupervised learning*, has no predefined labels, and part of the task is to determine what sort of structure would be suitable for the data [2]. This often takes the form of *clustering*, i.e., partitioning the data into groups in such a way that members of each group are somehow similar to each other. In addition, there are approaches between

supervised and unsupervised learning, including semi-supervised learning and reinforcement learning [27]. The focus of this thesis is, however, mainly on supervised learning.

To illustrate the use of some of the methods presented later, they will be applied to two well-known examples of machine learning tasks:

- The Boston housing data set [5] is a set with 14 attributes for 506 objects, and the modelling task is to predict the value of a house/apartment from the 13 other properties.

- The laser data known as Santa Fe A is selected from the Santa Fe Time Series Competition [110, 111]. The series contains 1000 samples of intensity data of a a far-infrared-laser in a chaotic state, and the task is to perform one-step-ahead prediction.

2.2 Supervised learning methods

2.2.1 Linear regression

Some of the simplest and most common models are realised in terms of a linear function of the input variables. *Linear regression* [11] is the model where the output label is approximated by a linear (or affine) function:

$$y_i \approx f(\boldsymbol{x}_i) = \sum_{j=1}^{d} w_j x_{i,j} + b \tag{2.1}$$

The parameters which define the model are the weights w_i and bias term b. The notation is often simplified by defining a new component $x_{i,0}$ which is 1 for all samples i, and equating $w_0 = b$, resulting in:

$$f(\boldsymbol{x}_i) = \sum_{j=1}^{d} w_j x_{i,j} + b = \sum_{j=0}^{d} w_j x_{i,j} = \boldsymbol{w}^T \boldsymbol{x}_i \tag{2.2}$$

Training this model is usually done by the method of *least squares*, finding those parameter values w_j which minimise the sum of squared errors between the labels and predictions on the training set [40]. Collecting the samples $\boldsymbol{x}_i$ as rows into a matrix $\mathbf{X}$ with the corresponding labels in the vector $\boldsymbol{y}$, this is expressed as

$$\boldsymbol{y} = \mathbf{X}\boldsymbol{w} + \boldsymbol{e}\,, \tag{2.3}$$

where $\boldsymbol{e}$ represents the modelling error. The least squares formulation corresponds to the optimisation problem of minimising this error:

$$\boldsymbol{w}_{\text{opt}} = \arg\min_{\boldsymbol{w}} \|\boldsymbol{e}\|^2 = \arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \mathbf{X}\boldsymbol{w}\|^2 \tag{2.4}$$

The optimum is found as the solution of the normal equations,

$$(\mathbf{X}^T\mathbf{X})\boldsymbol{w} = \mathbf{X}^T\boldsymbol{y}, \tag{2.5}$$

which in turn can be efficiently solved for $\boldsymbol{w}$ by any of a number of existing algorithms for solving linear systems of equations [12, 40]. In general, the the number of available samples $(\boldsymbol{x}_i, y_i)$ should exceed the number of variables, and the preferable situation is even that $N \gg d$ in order to reliably estimate the weights w_j. The case when $N < d$ can still be approached through *ridge regression* [50], where a penalty term is added to Eq. (2.4) in order to restrict the norm of $\boldsymbol{w}$. This is also known as *Tikhonov regularisation*.

Generalised linear model

A generalisation of the linear model in Eq. (2.1) is obtained by defining a fixed set of basis functions $\phi_j(\cdot)$, and modelling the output as a linear combination of these [11]:

$$f(\boldsymbol{x}_i) = \sum_{j=0}^{M} w_j \phi_j(\boldsymbol{x}_i) \tag{2.6}$$

An example of this is polynomial regression, where the functions ϕ_j take the form of different powers of the components of $\boldsymbol{x}_i$ and their products.

The model in Eq. (2.6) is still linear with respect to the parameters w_j, and can be solved by the method of least squares. In this case, the number of basis functions (M) should be less than N.

2.2.2 Method of nearest neighbours

A simple non-linear method for classification or regression is the nearest neighbour (NN) method [22, 94]. The idea is that given a query point to look through the dataset and find the closest point, then take the label of that point as the prediction:

$$y_i \approx y_{\text{NN}(i)} \quad \text{where} \quad \text{NN}(i) := \arg\min_{j \neq i} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|. \tag{2.7}$$

This is readily extended to k nearest neighbours (k-NN), where the prediction is the average of the k closest points. In regression, this is usually the arithmetic mean, whereas in classification tasks it is more suitable to use the mode.

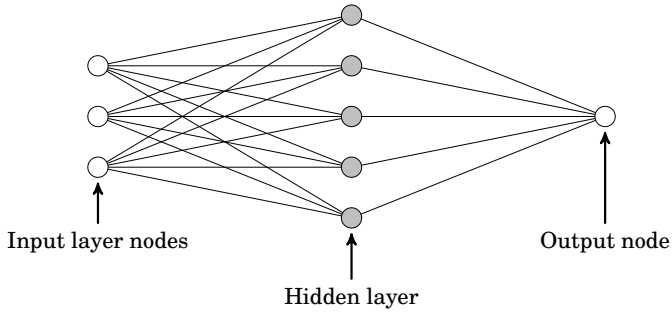

Figure 2.1. A single hidden-layer feed-forward neural network.

2.2.3 Neural networks and the Extreme Learning Machine

An *artificial neural network* [51, 27, 10] is a computational model inspired by the structure of the brain. Each neuron is a node which performs a processing task, such as combining several inputs and applying a function to produce an output value. After assigning the parameters to define the network, measured variables are mapped to the input nodes, and processing the network produces a result in the output node. A properly trained neural network can be a highly efficient and accurate prediction model.

A common type of neural network is the *multilayer perceptron* [27], referring to the topology of arranging the neurons in consecutive layers. A frequently used model is the single hidden-layer feed-forward neural network (Figure 2.1), which has only one layer between the input and output layers. Each input is connected to every node in the hidden layer, and the output is a linear combination of the hidden-layer neurons. The number of output nodes is determined by the number of values to predict for each input vector, and the output layer nodes could also be non-linear, if appropriate. For example, a network with d input nodes, one output node, and M neurons in the hidden layer can be written as

$$f(\boldsymbol{x}) = \sum_{k=1}^{M} \beta_k h\left(\sum_{j=1}^{d} w_{kj} x_j\right) \tag{2.8}$$

where $h(\cdot)$ is an appropriate activation function. The activation function is often chosen to be a continuous, bounded, and monotonous function to simulate the "spiking" of a neuron, where the neuron changes state from inactive to active when the input increases above a threshold value. Popular choices include the sigmoid and hyperbolic tangent functions [10]:

$$\sigma(t) = \frac{1}{1+\exp(-t)} \qquad \tanh(t) = \frac{1-\exp(-2t)}{1+\exp(-2t)} \tag{2.9}$$

While neural networks have potential to be powerful models, training

the network on data can be notoriously difficult. Training entails finding optimal values for all the weights w_{kj} and β_k. An effort to minimise the mean squared error using gradient descent leads to the common method of *back-propagation*, where all the weights are iteratively updated, taking a small step in the direction which most decreases the error on each step [49, 51, 27]. This approach can work well after proper training, but back-propagation can require a long time to converge. Other potential complications include converging to sub-optimal minima and *overtraining*. In a sufficiently large network, the algorithm can proceed to incorporate spurious properties of individual training samples if it runs for too long. This overtraining can be avoided by various early stopping criteria [27].

A recent development is the *Extreme Learning Machine* (ELM) [55], which is a single hidden-layer feed-forward neural network where *only* the output weights β_k are optimised, and all the weights w_{kj} between the input and hidden layer are assigned randomly.

$$\mathbf{H}\boldsymbol{\beta} = \boldsymbol{y} \quad \text{where} \quad H_{ik} = h\left(\boldsymbol{w}_k^T \boldsymbol{x}_i\right) \tag{2.10}$$

Training this model is much simpler, as the optimal output weights β_k can be calculated by ordinary least squares. The method relies on the idea of random projection: mapping the data randomly into a sufficiently high-dimensional space means that a linear model is likely to be relatively accurate. As such, the number of hidden-layer neurons needed for achieving equivalent accuracy is often much higher than in a multilayer perceptron trained by back-propagation, but the computational burden is still nearly negligible.

A high number of hidden layer neurons introduces concerns of overfitting, and regularised versions of the ELM have been developed to remedy this issue. These include the *optimally pruned ELM* (OP-ELM) [76], and its Tikhonov-regularised variant TROP-ELM [77].

2.2.4 Least squares support vector machines

One widely used non-linear model is *Least Squares Support Vector Machines* (LS-SVM) [100]. It is a variation of the original support vector machines [20], designed to be computationally lighter without sacrificing accuracy. The technique is closely related to that of *Gaussian processes* [84].

This section presents a brief summary of the method, see [100] for a

detailed exposition. The model can be represented in its primal space as

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\varphi}(\boldsymbol{x}) + b\,, \tag{2.11}$$

where $\boldsymbol{\varphi} : \mathbb{R}^d \to \mathbb{R}^{n_h}$ is a mapping to a higher dimensional *feature space* (possibly even infinite dimensional), $\boldsymbol{w}$ is a corresponding weight vector, and b a bias term. Training of the model is performed by the minimisation problem

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{e}} J(\boldsymbol{w},\boldsymbol{e}) &= \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{1}{2}\gamma \|\boldsymbol{e}\|^2 \\ \text{s.t.} \qquad y_i &= \boldsymbol{w}^T \boldsymbol{\varphi}(\boldsymbol{x}_i) + b + e_i \qquad i \in \{1,\ldots,N\} \end{aligned} \tag{2.12}$$

The function J is the sum of a regularisation term and the fitting error. The relative weights of the two terms, and the extent of the regularisation, is determined by the positive, real parameter γ. The problem is impractical in the primal space, since $\boldsymbol{\varphi}(\boldsymbol{x})$ and $\boldsymbol{w}$ are potentially infinite dimensional, and for this reason it is studied in the dual space, where $\boldsymbol{\varphi}(\boldsymbol{x})$ does not have to be explicitly constructed. Instead, it suffices to define a kernel K such that

$$K(\boldsymbol{x}_i,\boldsymbol{x}_j) = \boldsymbol{\varphi}(\boldsymbol{x}_i)^T \boldsymbol{\varphi}(\boldsymbol{x}_j) \tag{2.13}$$

With this, the model can be written as

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i K(\boldsymbol{x},\boldsymbol{x}_i) + b \tag{2.14}$$

and the parameters b and $\boldsymbol{\alpha}$ can be solved from a linear system. The most common choice for K is the radial basis function (Gaussian) kernel:

$$K(\boldsymbol{x}_i,\boldsymbol{x}_j) = \exp\left\{ -\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{\sigma^2} \right\}, \tag{2.15}$$

where the parameter σ determines the kernel width. Using LS-SVM with the RBF kernel then requires the user to choose two real valued parameters: γ and σ. The selection of these is non-trivial, and the parameters can not be optimised separately from each other. One suggested method to perform this tuning is by a grid search to minimise the random k-fold cross-validation error of the resulting model (Section 2.7).

2.3 Variable selection

In modern modelling problems, it is not uncommon to have an overwhelming number of input variables. Many of them may turn out to be irrelevant for the task at hand, but without external information it is often difficult

to identify these variables. *Variable selection* (also known as *feature selection*, *subset selection*, or *attribute selection* [80]) is the process of automating this task of choosing the most representative subset of variables for some modelling task.

Variable selection is a special case of dimensionality reduction. It can be used to simplify models by refining the data through discarding insignificant variables. As many regression models and other popular data analysis algorithms suffer from the so-called *curse of dimensionality* [7] to some degree it is necessary to perform some kind of dimensionality reduction to facilitate their effective use [107, 31, 66].

In contrast to general dimensional reduction techniques, variable selection provides additional value by distinctly specifying which variables are important and which are not [47]. This leads to a better intuitive insight into the relationship between the inputs and outputs, and assigns interpretability to the input variables. In cases where the user has control over some inputs, variable selection emphasises which variables to focus on and which are likely to be less relevant. Furthermore, discarding the less important inputs may result in cost savings in cases where measuring some properties would be expensive (such as chemical properties of a substance).

Variable selection techniques are in general based on either *variable ranking* or *subset selection* [47]. While subset selection methods attempt to return a single optimal subset of variables, the ranking methods only provide an ordering of the variables' estimated relevance for predicting the output. For regressions tasks, it is then left up to the user to select how many of the top ranked variables to choose. Due to their nature, ranking methods often fail to recognise situations where certain variables are useful only when combined with specific other variables. Subset selection is generally computationally more expensive, and the problem is known to be NP-hard even for a linear classifier [46, 3].

2.3.1 Correlation and linear methods

The simplest effective variable ranking method is to calculate and rank each input X_k by the *Pearson correlation coefficient* between it and the output Y [47, 48]:

$$\rho_k = \frac{\mathrm{Cov}(X_k, Y)}{\sigma_{X_k}\sigma_Y}. \tag{2.16}$$

The covariances and standard deviations can be estimated from data. This measure can only account for linear dependence of the output on the inputs, and is unable recognise more intricate connections between the variables. Using the correlation by itself is not ideal, since it does not account for correlations between the input variables, and is thus unable to detect redundant variables.

An improvement is to consider a linear model with L^1 regularisation on the weight vector:

$$\min_{\boldsymbol{w}} \|\boldsymbol{y} - \mathbf{X}\boldsymbol{w}\| \quad \text{s.t.} \quad \|\boldsymbol{w}\|_1 \leq t \tag{2.17}$$

This is known as the LASSO method (*least absolute shrinkage and selection operator*) [104]. The regularisation leads to solutions where many of the weights are zero, and it can be an effective form of variable selection by only considering the variables with non-zero weights for further study. Gradually increasing the value of t leads to new variables being included one by one.

Least angle regression (LARS) [28] is an efficient implementation of LASSO to solve the problem for all values of t. The order in which the variables are selected provides a ranking of their usefulness for predicting the output. Compared to the simple ranking by correlation, LARS is better as it specifically chooses the variables based on how much of the *residual* they can explain, i.e., how much *new* information they bring. This avoids the selection of an undesired variable in situations were a variable is highly correlated with the output only because it is highly correlated with some of the other highly correlated inputs.

As the method only ranks the input variables, it does not explicitly specify the number of top-ranked variables to select for optimal results, and this must somehow be chosen by the user.

2.3.2 Mutual information

The *mutual information* (MI) [23] is a measure of dependence between two random variables. It can be defined through the Shannon entropy:

$$\begin{aligned} I(X;Y) &= H(Y) - H(Y|X) \\ &= H(X) + H(Y) - H(X,Y) \end{aligned} \tag{2.18}$$

where the *entropy* H is the expected information content, which can be written in terms of a random variable's probability mass function $p_x(X)$:

$$H(X) = \mathrm{E}[-\log(p_x(X))] \tag{2.19}$$

If X is continuous, $H(X)$ as calculated above with the probability density function $p_x(X)$ is known as the *differential entropy* [23]. The conditional entropy $H(Y|X)$ and joint entropy $H(X,Y)$ are defined analogously through the conditional and joint probability distributions.

For continuous random variables X, Y with a joint distribution described by the density $p(x,y)$, the definition is equivalent to the integral below.

$$I(X;Y) = \int_Y \int_X p(x,y) \log\left(\frac{p(x,y)}{p_x(x)p_y(y)}\right) dx\, dy \tag{2.20}$$

Here p_x and p_y are the marginal probability densities of the random variables.

Another interpretation of mutual information is that it is the Kullback–Leibler (KL) divergence [23] of the product of the marginal distributions $p_x(x)p_y(y)$ from the joint distribution $p(x,y)$. The KL divergence is a measure of the difference between the distributions. If X and Y are independent random variables, the joint density is separable as the product $p(x,y) = p_x(x)p_y(y)$, and the divergence is 0. The more dependent the variables are, the larger the divergence, and the higher the value of the mutual information is.

Estimating mutual information

Early attempts to estimate mutual information from data with an unknown structure have been based on binning and histograms. That approach is not feasible for data with more than a couple of dimensions, as the number of samples required for reliable estimates grows exponentially. A general complication with using probability density estimation is that the tails of the distribution are particularly difficult to estimate accurately.

More recently, Kraskov et al. [63] proposed a method to estimate mutual information by considering nearest neighbours of each point in the input and output spaces separately and together. This approach has proven effective, and gained popularity.

Maximum Likelihood Mutual Information (MLMI) [101, 102] is another recent development promising accurate estimates.

Variable selection by mutual information

The seminal work on feature selection with mutual information [6] formulates the problem as follows: find the subset with k features that maximises the mutual information, for some a priori fixed value k. The MI is only estimated from histograms and binning.

For classification problems, a method involving kernel density estimation for the conditional distribution of each class has been used to estimate mutual information for variable selection [65], and later extended to dimensionality reduction [64].

A suggestion for regression problems is to find variables which maximise Kraskov's mutual information estimator [90]. To gauge the uncertainty of Kraskov's estimator, a resampling strategy has been proposed which can also help in determining how many variables to select in a forward search [32]. Extending variable selection to datasets with missing values, the partial distance strategy (PDS) has been used to find nearest neighbours for Kraskov's estimator [26].

Mutual information has also been used for a visualisation procedure for grouping features [39].

While optimising mutual information generally leads to accurate models in more concrete performance measures (classification rate, mean squared error), it has been shown that pathological examples exist where this is not true [34]. The adequacy of mutual information for estimating prediction accuracy is more precisely detailed in [33].

MI and mean squared error

While concerns have been raised over the use of MI as representative of prediction error [34, 33], there is a clear connection between the two measures. In the general case, MI implies a lower bound for the mean squared error (MSE) of an arbitrary estimator $\hat{Y}(X)$:

$$\mathrm{E}[(Y-\hat{Y}(X))^2] \geq \frac{1}{2\pi e} e^{2H(Y|X)} \tag{2.21}$$

[23, Thm. 8.6.6] when the entropy H is in nats (base e logarithm). Here equality is achievable only for the optimal estimator $\hat{Y}(X) = \mathrm{E}(Y|X)$ *and* if the residual $Y - \hat{Y}(X)$ is Gaussian.

Since $H(Y|X) = H(Y) - I(X;Y)$, it holds that that

$$\begin{aligned} \mathrm{E}[(Y-\hat{Y}(X))^2] &\geq \frac{1}{2\pi e} e^{2(H(Y)-I(X;Y))} \\ &= Ce^{-2I(X;Y)} \end{aligned} \tag{2.22}$$

where $C = \frac{1}{2\pi} e^{2H(Y)-1}$ is a constant that does not depend on the chosen variables X. Increasing the MI thus reduces the lowest achievable error.

2.3.3 The Relief algorithm

The Relief algorithm [61] is another popular method for feature selection for classification problems. The idea is based on evaluating variables

based on how well they distinguish samples close to the class boundary from each other. This is done by considering the *nearest hit* (nearest neighbour from the same class) and *nearest miss* (from a different class). If nearby misses have a large difference in the values of a certain variable, that variable is considered useful. On the other hand, variables with large differences for nearby hits are not as useful. The method has been extended to the ReliefF algorithm [88], which considers more than one nearest hit/miss for each class, among other efficiency improvements.

A variant for regression problems has also been introduced, called RReliefF [88]. As the concepts of hits and misses no longer apply when the target is continuous, they are replaced by a measure of how large the differences in the outputs of the nearest neighbours are, compared to differences in the input variables.

The output of the algorithm is a set of weights for the input variables, which can be converted to a ranking of the variables in order of importance. The Relief algorithm is able to effectively capture non-linear dependencies between the input and output variables. The main limitation is that since scores are given to each input variable individually, variables which are highly correlated with each other tend to all be selected, even when this is redundant.

2.4 Dealing with missing data

Most methods in machine learning are based on the assumption that data is available as a fixed set of measurements for each sample. This is not always true in practice, as several samples may have incomplete records for any of a number of reasons. These could include measurement error, device malfunction, operator failure, non-response in a survey, etc. Simply discarding the samples or variables which have missing components often means throwing out a large part of data that could be useful for the model. It is relevant to look for better ways of dealing with missing values in such cases.

In modelling such data, an assumption is that each missing value hides an underlying true value that is meaningful for analysis [69]. In the following, $\boldsymbol{x}_{\text{obs}}$ is the observed part of a data sample, $\boldsymbol{x}_{\text{mis}}$ is the true value of the unknown missing part, and M is a random variable indicating whether a certain value is missing or not. The vector $\boldsymbol{\theta}$ represents any other unknown parameters.

The cause for the data being missing is important to consider in order to approach the issue appropriately. Three categories of missingness mechanisms are generally identified [69]:

- *Missing completely at random* (MCAR), which is when the event of a value being missing is independent of any values, known or unknown:

$$p(M \mid \boldsymbol{x}_{\text{obs}}, \boldsymbol{x}_{\text{mis}}, \boldsymbol{\theta}) = p(M \mid \boldsymbol{\theta}) \tag{2.23}$$

- *Missing at random* (MAR) is the less restrictive situation where the missingness may depend on the value of the observed data:

$$p(M \mid \boldsymbol{x}_{\text{obs}}, \boldsymbol{x}_{\text{mis}}, \boldsymbol{\theta}) = p(M \mid \boldsymbol{x}_{\text{obs}}, \boldsymbol{\theta}) \tag{2.24}$$

  MAR is an ignorable missingness mechanism in the sense that maximum likelihood estimation still provides a consistent estimator [69].

- If the probability of having missing values depends on the missing values themselves, they are called *not missing at random* (NMAR). Modelling such data is generally only possible by introducing other, specific, assumptions, and will not be considered further in this thesis.

Two general procedures for handling missing values can be discussed separately from more specific models:

- *Complete-case analysis* implies discarding all incomplete samples, and using the remaining samples as if they constitute the entire data set. This simple approach is useful if abundant data is available since any standard analysis method can be applied without modification. Care should be taken to avoid biases, however, since if the data is not MCAR, the collection of complete samples is not a representative sampling of the entire data set.

- *Available-case analysis* is an alternative approach where for each individual estimate involving only a subset of variables, all samples with those variables present are included. For instance, statistics involving one (e.g., the mean) or two (e.g., covariance) variables can be estimated more accurately in this way. The disadvantage is that since different estimates are calculated using different sets of samples, combining them may lead to inconsistent results. For instance, consider estimating the

covariance of two variables by available-case analysis, and separately their variances as univariate available-case analyses. Using these to calculate the correlation coefficient in Eq. (2.16) could result in a value outside the allowed range from -1 to 1.

2.4.1 Imputation

Considering the assumption that every missing value represents an underlying true value, an intuitive approach is to consider filling in the missing value. This is known as *imputation*. It is a simple idea, but may not be that effective due to how errors propagate.

There are several paradigms for imputing missing data used in conjunction with machine learning methods [69].

- *Conditional mean imputation* implies filling in the missing values by the best guess. This is optimal in terms of minimising the mean squared error of the imputed values, but suffers from leading to biased derived statistics of the data. For instance, estimates of variance or distances are negatively biased.

- *Random draw imputation* is more appropriate for generating a representative example of a fully imputed data set, but may have too much variability in estimates of any single values to be accurate.

- *Multiple imputation* is drawing several representative imputations of the data, analysing each set separately, and combining the results [91]. This can result in unbiased and accurate estimates after a sufficiently high number of draws, but it is not always straightforward to determine the posterior distribution to draw from [29, 92]. In the context of machine learning, repeating the analysis several times is however impractical as training and analysing a sophisticated model tends to be computationally expensive.

If the fraction of missing data is sufficiently small, a practical preprocessing step is to take any reasonable imputation method to fill in the missing values and proceed with conventional methods for further processing. Any errors introduced by inaccurate imputation may be considered insignificant in terms of the entire processing chain. With a larger

proportion of measurements being missing, errors caused by the imputation are increasingly relevant, see, e.g., [30] for an analysis on the effect of imputation on classification accuracy.

A simple method of imputation by searching for the nearest neighbour among only the fully known patterns can be effective when only a few values are missing [53, 59, 18], but is ineffective when a majority of the data samples have missing components as the availability of candidates decreases rapidly. An improved approach is incomplete-case k-NN imputation (ICkNNI) [106], which searches for neighbours among all patterns for which a superset of the known components of the query point are known. This still fails in high-dimensional cases, or with a sufficiently large proportion of missing data. A more intricate method where multiple nearest neighbours are considered, and a model is separately learned for each incomplete sample, is presented in [109].

2.4.2 Estimating distances

The problem of directly estimating pairwise distances between samples with missing values is less studied. Previous approaches involve imputing the missing data with some estimates, and calculating distances from the imputed data. This technique severely underestimates the uncertainty of the imputed values. Estimating the distances directly leads to more reliable estimates as the uncertainty can also be considered.

A simple and somewhat widely used method for estimating distance with missing values is the Partial Distance Strategy (PDS) [25, 52]. In the PDS, an estimate for the squared distance is found by calculating the sum of squared differences of the mutually known components, and scaling the value proportionally to account for the missing values.

$$\hat{d}(\boldsymbol{x}_i,\boldsymbol{x}_j)^2 = \frac{d}{|O_i \cap O_j|} \sum_{l \in O_i \cap O_j} \left(x_{i,l} - x_{j,l}\right)^2. \tag{2.25}$$

The index sets O_i and O_j represent the observed components of the samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, respectively. As the contribution of the missing values is ignored even if the corresponding variable for the other sample is known, the accuracy of the method is limited and there is a tendency to exaggerate the variability of distances. In a nearest neighbour search, for instance, this manifests as a risk of returning samples with several missing values only because the few mutually known variables have similar values. Furthermore, if two samples have no common components, the output of this strategy is undefined. The PDS has nevertheless been used

to find nearest neighbours in order to estimate mutual information [26].

In a specific case of an entropy-based distance measure [19], the authors propose that the distance to an incomplete sample can be estimated as the mean distance after the missing values are replaced by random draws. However, the missing value is successively replaced by the corresponding attribute from every specified sample, ignoring any dependence to the observed attributes of the incomplete sample.

Finding distances from each sample to some prototype patterns (where the prototypes have no missing values) has been conducted by ignoring those components which are missing for the query pattern. Such distances from the same query point to different prototypes are comparable, and this strategy has, for instance, been used successfully with self-organising maps (SOM) [21]. This is, however, equivalent to the partial distance strategy, and suffers from the same limitations.

2.4.3 Methods to account for missing data intrinsically

For some machine learning methods, it is possible to use the incomplete samples in training the model without additional processing. One possibility for integrating the imputation of missing values with building a prediction model is presented in the MLEM2 rule induction algorithm [41]. A variation is to restrict the search to certain samples or attributes according to specified rules, as in the "concept closest fit" [41] and "rough sets fit" [67] methods.

Another suggested alternative is to use nearest neighbours to simultaneously conduct classification and imputation [36].

A way to use mixtures of Gaussians for training neural networks on data with missing values has previously been proposed in [105], involving finding the average gradient of the relevant parameters by integrating over the conditional distribution of missing values. However, the authors only specify widths for the Gaussian components separately for each dimension in their implementation. This simplifies the analysis greatly, effectively ignoring correlations by restricting the covariance matrices to be diagonal. The suggested procedure specifically applies to training the network by back-propagation, and cannot directly be used for other machine learning methods. Another more limited approach to directly allow incomplete samples to be used in back-propagation is to flag input neurons corresponding to unknown attributes as protected, temporarily restricting them from being modified [108]. Further suggested approaches to using

a mixture of Gaussians to model the input density for machine learning include forming hidden Markov models for speech recognition by integrating over the density [78]. Another analysis accounting for the uncertainty of missing values using a single multivariate Gaussian in clinical trials is [9].

2.5 Gaussian mixtures models

Mixtures of Gaussians can be used for a variety of applications by estimating the density of data samples [11]. A Gaussian mixture model is defined by its parameters. These consist of the mixing coefficients π_k, the means $\boldsymbol{\mu}_k$, and covariance matrices $\boldsymbol{\Sigma}_k$ for each component k ($1 \leq k \leq K$) in a mixture of K components. The combination of parameters is represented as $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$.

The model specifies a distribution in $\mathbb{R}^d$, given by the probability density function

$$p(\boldsymbol{x} \mid \boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{2.26}$$

where $\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the probability density function of the multivariate normal distribution

$$\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\right). \tag{2.27}$$

2.5.1 The EM algorithm

The standard procedure for fitting a Gaussian mixture to a data set is maximum likelihood estimation by the Expectation–Maximisation (EM) algorithm [24, 74]. The log-likelihood of a model given data $\mathbf{X}$ is

$$\log \mathcal{L}(\boldsymbol{\theta}; \mathbf{X}) = \log p(\mathbf{X} \mid \boldsymbol{\theta}) = \sum_{i=1}^{N} \log\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right), \tag{2.28}$$

where $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$ is the set of parameters to be determined.

As explicitly optimising Eq. (2.28) is difficult, $\mathbf{Z}$ is introduced as a set of latent binary variables z_{ik}, each representing whether sample i belongs to component k. A sample can belong to only one component, so for a fixed i, exactly one of z_{ik} for different k is non-zero. Write the complete data

log-likelihood as follows:

$$\log \mathcal{L}_C(\boldsymbol{\theta};\mathbf{X},\mathbf{Z}) = \log p(\mathbf{X},\mathbf{Z} \mid \boldsymbol{\theta}) \tag{2.29}$$

$$= \log \prod_{i=1}^{N} p(\boldsymbol{x}_i, \boldsymbol{z}_i \mid \boldsymbol{\theta}) \tag{2.30}$$

$$= \log \prod_{i=1}^{N} p(\boldsymbol{z}_i) p(\boldsymbol{x}_i \mid \boldsymbol{z}_i, \boldsymbol{\theta}) \tag{2.31}$$

$$= \log \prod_{i=1}^{N} \left(\prod_{k=1}^{K} \pi_k^{z_{ik}} \right) \left(\prod_{k=1}^{K} \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\theta}_k)^{z_{ik}} \right) \tag{2.32}$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{K} z_{ik} \log(\pi_k \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\theta}_k)) \tag{2.33}$$

The E-step is to find the expected value of the complete data log-likelihood function, with respect to the conditional distribution of latent variables $\mathbf{Z}$ given the data $\mathbf{X}$ under the current estimate of the parameters $\boldsymbol{\theta}^{(t)}$:

$$Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}) = \mathrm{E}\left[\log \mathcal{L}_C(\boldsymbol{\theta};\mathbf{X},\mathbf{Z}) \mid \mathbf{X}, \boldsymbol{\theta}^{(t)} \right] \tag{2.34}$$

Defining $t_{ik} = \mathrm{E}[z_{ik} \mid \mathbf{X}, \boldsymbol{\theta}^{(t)}]$, this reduces to

$$Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}) = \sum_{i=1}^{N} \sum_{k=1}^{K} t_{ik} \log(\pi_k \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\theta}_k)), \tag{2.35}$$

where t_{ik} for each sample $\boldsymbol{x}_i$ and component k is the probability that the sample belongs to that component. Given the current parameter estimates $\boldsymbol{\theta}^{(t)}$, it is calculated as

$$t_{ik} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\theta}_k^{(t)})}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\theta}_j^{(t)})}. \tag{2.36}$$

In the M-step, the expected log-likelihood is maximised:

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}), \tag{2.37}$$

which corresponds to re-estimating the model parameters using the updated probabilities:

$$N_k = \sum_{i=1}^{N} t_{ik}, \tag{2.38}$$

$$\pi_k = \frac{N_k}{N}, \tag{2.39}$$

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N} t_{ik} \boldsymbol{x}_i, \tag{2.40}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{i=1}^{N} t_{ik} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T. \tag{2.41}$$

In a practical implementation, the E-step in Eq. (2.36) and M-step in Eqs. (2.38)–(2.41) are alternated until convergence is observed in the log-likelihood. The initialisation before the first E-step is arbitrary. The clustering algorithm K-means is a popular choice to find a reasonable initialisation [11].

2.5.2 With missing values

A Gaussian mixture model with the EM algorithm works well on incomplete data sets, since any missing values can be included in the same framework [37, 56].

The data $\mathbf{X}$ now contains the observations $\{\boldsymbol{x}_i\}_{i=1}^N$ so that for each sample i there is an associated index set $O_i \subseteq \{1,\ldots,d\}$ representing which variables are known (observed). The complement set M_i corresponds to the missing values for that sample. The observed part of the full data is referred to by $\mathbf{X}^O$, and the observed data log-likelihood with missing values is

$$\log \mathscr{L}(\boldsymbol{\theta};\mathbf{X}^O) = \log p(\mathbf{X}^O \mid \boldsymbol{\theta}) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}\left(\boldsymbol{x}_i^{O_i} \,\middle|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right) \right) \tag{2.42}$$

where as a shorthand of notation, $\mathcal{N}(\boldsymbol{x}_i^{O_i} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is also used for the *marginal* multivariate normal distribution probability density of the observed values of a sample $\boldsymbol{x}_i$.

In order to apply the EM algorithm, additional latent variables $\mathbf{X}^M$ are introduced for all the missing values in the data. The E-step is then to find

$$Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}) = \mathrm{E}\left[\log \mathscr{L}_C(\boldsymbol{\theta};\mathbf{X}^O,\mathbf{X}^M,\mathbf{Z}) \,\middle|\, \mathbf{X}^O, \boldsymbol{\theta}^{(t)} \right] \tag{2.43}$$

where the expectation is with respect to all the variables in both $\mathbf{X}^M$ and $\mathbf{Z}$ (the unknown component memberships).

This requires some additional computation, including the conditional expectations of the missing components of a sample with respect to each Gaussian component k, and their conditional covariance matrices, i.e.,

$$\tilde{\mu}_{ik}^{M_i} = \mathrm{E}\left[\boldsymbol{x}_i^{M_i} \,\middle|\, \boldsymbol{x}_i^{O_i}, z_{ik} = 1 \right] \tag{2.44}$$

$$\tilde{\boldsymbol{\Sigma}}_{ik}^{MM_i} = \mathrm{Cov}\left[\boldsymbol{x}_i^{M_i} \,\middle|\, \boldsymbol{x}_i^{O_i}, z_{ik} = 1 \right] \tag{2.45}$$

where the statistics are conditional on the assumption that $\boldsymbol{x}_i$ originates from the kth Gaussian.

Then the E-step is:

$$t_{ik} = \frac{\pi_k \mathcal{N}\left(\boldsymbol{x}_i^{O_i} \,\middle|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(\boldsymbol{x}_i^{O_i} \,\middle|\, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)}, \tag{2.46}$$

$$\tilde{\boldsymbol{\mu}}_{ik}^{M_i} = \boldsymbol{\mu}_k^{M_i} + \boldsymbol{\Sigma}_k^{MO_i} \left(\boldsymbol{\Sigma}_k^{OO_i}\right)^{-1} \left(\boldsymbol{x}_i^{O_i} - \boldsymbol{\mu}_k^{O_i}\right), \tag{2.47}$$

$$\tilde{\boldsymbol{\Sigma}}_{ik}^{MM_i} = \boldsymbol{\Sigma}_k^{MM_i} - \boldsymbol{\Sigma}_k^{MO_i} \left(\boldsymbol{\Sigma}_k^{OO_i}\right)^{-1} \boldsymbol{\Sigma}_k^{OM_i}, \tag{2.48}$$

For convenience, also define corresponding imputed data vectors $\tilde{\boldsymbol{x}}_{ik}$ and full covariance matrices $\tilde{\boldsymbol{\Sigma}}_{ik}$ which are padded with zeros for the known

components.

$$\tilde{\boldsymbol{x}}_{ik} = \begin{pmatrix} \boldsymbol{x}_i^{O_i} \\ \tilde{\boldsymbol{\mu}}_{ik}^{M_i} \end{pmatrix}, \tag{2.49}$$

$$\tilde{\boldsymbol{\Sigma}}_{ik} = \begin{pmatrix} \mathbf{0}^{OO_i} & \mathbf{0}^{OM_i} \\ \mathbf{0}^{MO_i} & \tilde{\boldsymbol{\Sigma}}_{ik}^{MM_i} \end{pmatrix}. \tag{2.50}$$

The M-step is nearly the same as with fully observed data, only using the imputed samples for the calculations, and including the conditional covariances for the imputed values.

$$N_k = \sum_{i=1}^{N} t_{ik}, \tag{2.51}$$

$$\pi_k = \frac{N_k}{N}, \tag{2.52}$$

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N} t_{ik} \tilde{\boldsymbol{x}}_{ik}, \tag{2.53}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{i=1}^{N} t_{ik} \left[(\tilde{\boldsymbol{x}}_{ik} - \boldsymbol{\mu}_k)(\tilde{\boldsymbol{x}}_{ik} - \boldsymbol{\mu}_k)^T + \tilde{\boldsymbol{\Sigma}}_{ik} \right], \tag{2.54}$$

2.5.3 Model selection

When using the EM algorithm to fit a mixture model, the number of components K must be fixed beforehand. This selection is crucial and has a significant effect on the resulting accuracy. Too few components are not able to model the distribution appropriately, while having too many components can cause overfitting.

The number of components can be selected according to the Akaike information criterion (AIC) [1] or the Bayesian information criterion (BIC) [93]. Both are expressed as a function of the log-likelihood of the converged mixture model:

$$\text{AIC} = -2 \log \mathscr{L}(\boldsymbol{\theta}; \mathbf{X}) + 2P, \tag{2.55}$$

$$\text{BIC} = -2 \log \mathscr{L}(\boldsymbol{\theta}; \mathbf{X}) + \log(N) P, \tag{2.56}$$

where $P = Kd + \frac{1}{2}Kd(d+1) + K - 1$ is the number of free parameters. In practice, the EM algorithm is run separately for several different values of K, and the model which minimises the chosen criterion is selected. As $\log(N) > 2$ in most cases, BIC more aggressively penalises an increase in P, generally resulting in a smaller choice for K than by AIC.

Another choice is the Akaike information criterion [1] with the small sample (second-order) bias adjustment [57]. Using the corrected version

can be useful, as the number of parameters grows relatively fast (quadratically) when increasing the number of components.

$$\mathrm{AIC}_C = -2\log\mathscr{L}(\boldsymbol{\theta};\mathbf{X}) + 2P + \frac{2P(P+1)}{N-P-1} \tag{2.57}$$

With high-dimensional data sets, the number of parameters quickly tends to become larger than the number of available samples when increasing the number of components, and the criterion would not be valid anymore. This effect can be mitigated by imposing restrictions on the structure of the covariance matrices, but this would also make the model less powerful.

Minimum description length [87, 42] is a general principle for model selection. In the current case of choosing the number of components, it is equivalent to BIC in Eq. (2.56) above [103]. Several further criteria are discussed in [75, Ch. 6].

2.5.4 High-dimensional data

As the number of free parameters grows with the square of the data dimension, in high-dimensional cases it is often not possible to fit a conventional Gaussian mixture model, or even a model with a single Gaussian component. A practical solution is that the covariance matrices are restricted to being identical [75]

$$\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma} \quad \forall k \tag{2.58}$$

Alternatively, each covariance matrix could be forced to zero for all off-diagonal elements, although this has the side effect of aligning the components along the coordinate axes. The further simplification where the variance is equal in all directions, $\boldsymbol{\Sigma}_k = \sigma_k^2\mathbf{I}$, is hence often preferred as it retains the rotational invariance.

Another possibility is *high-dimensional data clustering (HDDC)* [13], which is a Gaussian mixture model where the covariance matrices are replaced by a reduced representation. From the different variants of HDDC, the basic version allow all the parameters of the reduced representation to be determined freely for each component. In brief (see [13] for details and derivations), the reduced representation entails taking the eigenvalue decomposition of the covariance matrix

$$\boldsymbol{\Sigma}_k = \mathbf{Q}_k\boldsymbol{\Lambda}_k\mathbf{Q}_k^T \tag{2.59}$$

and modifying all the eigenvalues λ_{kj} apart from some of the largest ones. Only the d_k largest eigenvalues are kept exactly, and the remaining ones

are replaced by their arithmetic mean. In other words, the covariance matrix $\boldsymbol{\Sigma}_k$ is replaced by a matrix $\boldsymbol{\Sigma}'_k = \mathbf{Q}_k \boldsymbol{\Lambda}'_k \mathbf{Q}_k^T$ where $\boldsymbol{\Lambda}'_k$ is a diagonal matrix with the elements

$$\lambda'_{kj} = \begin{cases} \lambda_{kj} & j \leq d_k \\ b_k & j > d_k \end{cases} \tag{2.60}$$

where

$$b_k = \frac{1}{d - d_k} \sum_{l=d_k+1}^{d} \lambda_{kl} = \frac{1}{d - d_k} \left(\mathrm{tr}(\boldsymbol{\Sigma}_k) - \sum_{l=1}^{d_k} \lambda_{kl} \right), \tag{2.61}$$

assuming the eigenvalues λ_{kj} are in decreasing order. This representation implies that only the first d_k eigenvalues and eigenvectors need to be calculated and stored, efficiently reducing the number of free parameters required to specify each Gaussian component. The number of significant eigenvalues can be determined by the scree test [15], where the dimension is selected when the subsequent eigenvalues have a difference smaller than a specified threshold.

Applying this idea to the case of missing data is possible by modifying the covariance matrices of each component after calculating them in the M-step in Eqs. (2.51)–(2.54). However, the computational gains obtained from having a reduced representation are not available, as the full covariance matrices still need to be inverted in order to calculate the conditional parameters in the following E-step.

2.6 Time series analysis and modelling

A time series is one of the most common forms of data, and has been studied extensively from weather patterns spanning centuries to sensors and microcontrollers operating on nanosecond scales. A time series is any sequence of numbers where the order corresponds to the temporal order [14]. Typically this is a quantity that is measured at regular intervals, such as end of day stock prices, yearly rainfall in an area, or the average number of sunspots visible each month. From a machine learning perspective, the most relevant tasks tend to be prediction of one or several future data points, or interpolation to fill in gaps in the data.

In order to conduct any meaningful analysis, the values of a time series must follow some underlying rules connecting them to previous values. A typical assumption is that a time series is stationary, i.e., the parameters governing the generative model do not change over time. This enables the

possibility of estimating the relevant parameters from a sufficiently long sample of the time series.

Many types of time series can be modelled by linear methods, such as auto-regressive (AR) models [14]. For example, an autoregressive model of order p, AR(p), of a time series z_t would be represented as

$$z_t = \sum_{j=1}^{p} w_j z_{t-j} + \varepsilon_t, \tag{2.62}$$

where the current value z_t is a linear combination of previous terms with additive noise ε_t. This expression is directly usable as a prediction model.

Some time series are not adequately explainable by linear models, but can instead be modelled by non-linear regression analysis, where the current value is a non-linear function of previous values [60, 38].

2.7 Model selection, evaluation and parameter optimisation

Model selection refers to the process of finding a model structure which most appropriately fits the available data. In many cases, the structure of a model can be parametrised in terms of some hyperparameters, and several choices for these parameters can be evaluated according to the resulting fitting error. The parameter set giving the best result is then selected.

An important concern in machine learning is the *generalisation* ability of a model, referring to how accurately the model can handle previously unseen data samples [2]. Given any training data, it is easy to find a perfectly fitting model simply by making it complex enough, but such a model is unlikely to be able to deal with new data properly. The training data is a representative sample of a more general distribution, and a good model should only take into account those properties which are relevant in the general case.

The crucial step of evaluating the performance of a model is not an obvious issue. For many models, it is sensible to examine the output error $y_i - f(\boldsymbol{x}_i)$ for each sample, and take the average of the square of these to obtain the *mean squared error* (MSE). When the same data is used both for building the model and evaluation, this is known as the *training error*. The error measure that the model produces on new data from the underlying distribution is the *generalisation error* [79]. In many cases, the training error significantly underestimates the generalisation error, as the model is optimised on the same data as it is evaluated on. This is

an example of *overfitting*.

The origin of overfitting can be understood through the bias-variance decomposition of the expected error. In the following, assume a regression model $\hat{f}(X)$ is built on a dataset originating from $Y = f(X) + \varepsilon$, where the noise ε is zero mean and independent from X. The expected squared error at a point x_0 can be decomposed as follows:

$$\mathrm{E}\left[(Y - \hat{f}(x_0))^2 \mid X = x_0\right] = \underbrace{\mathrm{E}\left[(Y - f(x_0))^2 \mid X = x_0\right]}_{\text{noise}} + \underbrace{\left(f(x_0) - \mathrm{E}[\hat{f}(x_0)]\right)^2}_{(\text{bias})^2} + \underbrace{\mathrm{E}\left[(\hat{f}(x_0) - \mathrm{E}[\hat{f}(x_0)])^2\right]}_{\text{variance}} \quad (2.63)$$

Here the expectation of the model should be interpreted as taken over the distribution of different datasets on which the model is built, i.e., realisations of datasets $\{(x_i, y_i)\}_{i=1}^{N}$. The error resulting from the noise ε is unavoidable. The second term is the bias of the model as an estimator of the true function; this is minimised by fitting the model to the data. The last term is the variance of the model around its mean, representing how sensitive it is to differences between realisations of the data. Typically, making a model more complex has the intended outcome of decreasing the bias, but also the undesired effect of increasing the variance. This dilemma is known as the *bias-variance trade-off*, and overfitting occurs when the increase in variance exceeds the improved fit of the model.

To recognise and avoid overfitting, the samples which are used for training and evaluation must be separated. The available data can be split into two complementary sets, the *training* and *test* sets [2]. If the model is trained on the training set and evaluated on the test set, the resulting MSE is likely to be a better indicator of the generalisation error. Often, a separate *validation* set is additionally partitioned for parameter selection. Building several models with different parameter values, their relative performance can be assessed on the validation set, and the best selected. A typical rule of thumb for splitting the data is 50% for the training set, and 25% each for validation and testing [50].

The process of splitting into training and validation sets can be performed repeatedly to increase the confidence of the estimates. A common way to structure the repetition is *k-fold cross-validation* [79, 50], where the data is (usually randomly) partitioned into k equally sized sets. Each of the k sets is sequentially chosen to be the test set, and the model is trained on the union of the remaining $k-1$ sets. Averaging these test errors then provides a reasonable estimate for the generalisation error, as every sample has been used for testing exactly once.

A special case of cross-validation when $k = N$ is called *leave-one-out* (LOO) cross-validation [50]. As the name implies, here each single sample is sequentially left for the test set while the model is trained on the remaining samples, and the squared errors are averaged. As this generally requires the training of N models, it is often too inefficient to be practical, but for certain methods (such as the LS-SVM) it is possible to obtain the LOO error exactly without explicitly performing the repeated training of the model [16].

Another issue related to the evaluation of machine learning models is that the distribution of data in the intended final application may differ somewhat from the available training data. This is known as *dataset shift* [83], and if such changes can be expected, they should be taken into account in the model selection procedure. An overview of approaches to deal with different forms of dataset shift is presented in [83].

3. Contributions to Missing Data Methods

3.1 Distance estimation

In pattern recognition, it is mostly the distances between data points that matter. Many computational methods can be formulated in terms of pairwise distances between samples, or alternatively the distances between the samples and a set of prototypes. Nearest neighbours (k-NN) [94] and multidimensional scaling (MDS) [17] directly use the distances. Kernel-based methods, such as support vector machines (SVM) [20, 51], are usually applied with kernel matrices calculated from the pairwise distance matrix. Distances from samples to prototypes are also used in radial basis function (RBF) neural networks [10, 51] and self-organising maps (SOM) [62, 89, 51].

In most cases, the distance measure used is the Euclidean metric: given two samples, take the square-root of the sum of squared elementwise differences. With specified values for all elements of both data vectors, this is straightforward arithmetic. But if one or more of the elements are missing, the distance between the samples is a far more nebulous concept.

Some approaches to deal with the issue have been suggested – including calculating the distances after imputation, and the partial distance strategy – but these have limited accuracy. Specifically, while minimising the imputation error is a reasonable strategy for filling the data, calculating distances on the resulting data leads to a suboptimal estimate of the *distance*.

A procedure to estimate all pairwise distances in a data set immediately enables the use of any of the aforementioned machine learning techniques without having to consider any further tricks to deal with the missing values.

3.1.1 The expected squared distance

Given two arbitrary data vectors $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^d$, which may contain missing values, the target in Publication I is to estimate the *squared* Euclidean distance between them.

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j)^2 = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 = \sum_{l=1}^{d} (x_{i,l} - x_{j,l})^2 \tag{3.1}$$

The reasons for working with the squared distance directly originate from the observation that it is considerably easier to deal with than the non-squared distances. In addition, many methods specifically use the squared distance (e.g., RBF and SVM). In other cases where distances are only going to be sorted or ranked, such as in nearest neighbours, using the squared distance is equivalent.

The sum of squared differences can be partitioned into four parts depending on the missing and observed parts of each sample:

$$\begin{aligned}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 = \sum_{l=1}^{d} (x_{i,l} - x_{j,l})^2 = &\sum_{l \in O_i \cap O_j} (x_{i,l} - x_{j,l})^2 + \sum_{l \in O_i \cap M_j} (x_{i,l} - x_{j,l})^2 \\ &+ \sum_{l \in M_i \cap O_j} (x_{i,l} - x_{j,l})^2 + \sum_{l \in M_i \cap M_j} (x_{i,l} - x_{j,l})^2. \end{aligned} \tag{3.2}$$

The index sets O_i and O_j represent the observed components of the samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, respectively, and M_i and M_j correspondingly the missing values. The first term in the expression above ($l \in O_i \cap O_j$) includes those components which are known for both samples, and can be calculated directly. The remaining sums contain those parts where one or both of the values are missing. The missing values can be replaced with random variables $X_{i,l}$ for every $l \in M_i$. Taking the expected value of the expression and using the linearity of expectation leads to:

$$\begin{aligned}\mathrm{E}\left[\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right] = &\sum_{l \in O_i \cap O_j} (x_{i,l} - x_{j,l})^2 + \sum_{l \in O_i \cap M_j} \mathrm{E}[(x_{i,l} - X_{j,l})^2] \\ &+ \sum_{l \in M_i \cap O_j} \mathrm{E}[(X_{i,l} - x_{j,l})^2] + \sum_{l \in M_i \cap M_j} \mathrm{E}[(X_{i,l} - X_{j,l})^2] \end{aligned} \tag{3.3}$$

This further simplifies as

$$\begin{aligned}\mathrm{E}\left[\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right] = &\sum_{l \in O_i \cap O_j} (x_{i,l} - x_{j,l})^2 + \sum_{l \in O_i \cap M_j} \left((x_{i,l} - \mathrm{E}[X_{j,l}])^2 + \mathrm{Var}[X_{j,l}]\right) \\ &+ \sum_{l \in M_i \cap O_j} \left((\mathrm{E}[X_{i,l}] - x_{j,l})^2 + \mathrm{Var}[X_{i,l}]\right) \\ &+ \sum_{l \in M_i \cap M_j} \left((\mathrm{E}[X_{i,l}] - \mathrm{E}[X_{j,l}])^2 + \mathrm{Var}[X_{i,l}] + \mathrm{Var}[X_{j,l}]\right) \end{aligned} \tag{3.4}$$

In more detail, the second summation ($l \in O_i \cap M_j$) is expanded as

$$\begin{aligned}
\mathrm{E}[(x_{i,l} - X_{j,l})^2] &= \mathrm{E}[x_{i,l}^2 - 2x_{i,l}X_{j,l} + X_{j,l}^2] = x_{i,l}^2 - 2x_{i,l}\,\mathrm{E}[X_{j,l}] + \mathrm{E}[X_{j,l}^2] \\
&= x_{i,l}^2 - 2x_{i,l}\,\mathrm{E}[X_{j,l}] + \mathrm{E}[X_{j,l}]^2 - \mathrm{E}[X_{j,l}]^2 + \mathrm{E}[X_{j,l}^2] \\
&= (x_{i,l} - \mathrm{E}[X_{j,l}])^2 + \mathrm{E}[X_{j,l}^2 - \mathrm{E}[X_{j,l}]^2] \\
&= (x_{i,l} - \mathrm{E}[X_{j,l}])^2 + \mathrm{Var}[X_{j,l}]
\end{aligned}$$

The other cases are similar. The only assumption here is that in the final term where both observations are missing ($l \in M_i \cap M_j$), the random variables $X_{i,l}$ and $X_{j,l}$ are uncorrelated, given the known values of the samples.

It is important to note that no assumptions have been made about the distribution of the samples. The random variables $X_{i,l}$ for any individual sample can have arbitrary distributions, since the expression for the squared distance is linearly separable into the contributions of each dimension separately. The only information needed to calculate the expectation is the mean and variance of each missing value individually.

A general assumption in machine learning is that data samples are independent draws from some underlying multivariate probability distribution. Then the distributions of $X_{i,l}$ should be seen as the conditional distribution when the known values are fixed, with respect to the underlying distribution. Let the distribution consist of the set of random variable $\{X_l\}_{l=1}^{d}$ with a probability density $p(X_1,\ldots,X_l)$. Then $p(X_{i,l}) = p(X_l \mid \boldsymbol{x}_i^{O_i})$, and the expectations and variances above can be determined as the corresponding conditional expectations.

Now construct an imputed version of $\boldsymbol{x}_i$ and call it $\tilde{\boldsymbol{x}}_i$. Each missing value has been replaced by its conditional mean

$$\tilde{x}_{i,l} = \begin{cases} \mathrm{E}[X_l \mid \boldsymbol{x}_i^{O_i}] & \text{if } l \in M_i, \\ x_{i,l} & \text{otherwise} \end{cases} \tag{3.5}$$

Let $\sigma_{i,l}^2$ be the corresponding conditional variance

$$\sigma_{i,l}^2 = \begin{cases} \mathrm{Var}[X_l \mid \boldsymbol{x}_i^{O_i}] & \text{if } l \in M_i, \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

Using these, the result in Eq. (3.4) is conveniently written as

$$\mathrm{E}\left[\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right] = \sum_{l=1}^{d} \left((\tilde{x}_{i,l} - \tilde{x}_{j,l})^2 + \sigma_{i,l}^2 + \sigma_{j,l}^2\right) \tag{3.7}$$

An alternative form is also:

$$\mathrm{E}\left[\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right] = \|\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_j\|^2 + s_i^2 + s_j^2, \quad \text{where} \quad s_i^2 = \sum_{l \in M_i} \sigma_{i,l}^2 \tag{3.8}$$

Writing the expression like this shows how the uncertainty related to the missing values leads to an expected increase in the distance. Using an imputation method and then calculating the distance on the filled in data only accounts for the first term in this equation. Explicitly including the variance term is essential for an accurate estimate.

3.1.2 Using a multivariate normal distribution

The underlying multivariate distribution is usually not known, so the problem then becomes how to calculate the conditional statistics. Assume it is possible to estimate the first and second moments of the distribution from the data, i.e., the mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. If this is all the information available, it is reasonable to apply the principle of maximum entropy which states that the most appropriate model is the one that maximises the entropy (while satisfying modelling constraints). The distribution with maximal entropy for a given mean and covariance structure is the multivariate normal distribution with those parameters [23, Thm. 8.6.5].

Proceeding with the assumption that the data originates from a multivariate normal distribution with known parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, the conditional means and variances for Eqs. (3.5) and (3.6) are straightforward to calculate.

Let the d-dimensional random variable X be split into two parts according the missing and observed parts of a sample $\boldsymbol{x}_i$, and also partition the mean and covariance accordingly:

$$X = \begin{bmatrix} X^{M_i} \\ X^{O_i} \end{bmatrix}, \quad \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}^{M_i} \\ \boldsymbol{\mu}^{O_i} \end{bmatrix}, \quad \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}^{MM_i} & \boldsymbol{\Sigma}^{MO_i} \\ \boldsymbol{\Sigma}^{OM_i} & \boldsymbol{\Sigma}^{OO_i} \end{bmatrix} \tag{3.9}$$

It holds that the conditional distribution of X^{M_i} given $X^{O_i} = \boldsymbol{x}_i^{O_i}$ also follows a normal distribution, with mean

$$\tilde{\boldsymbol{\mu}}_i^{M_i} = \boldsymbol{\mu}^{M_i} + \boldsymbol{\Sigma}^{MO_i}\left(\boldsymbol{\Sigma}^{OO_i}\right)^{-1}\left(\boldsymbol{x}_i^{O_i} - \boldsymbol{\mu}^{O_i}\right) \tag{3.10}$$

and covariance matrix

$$\tilde{\boldsymbol{\Sigma}}_i^{MM_i} = \boldsymbol{\Sigma}^{MM_i} - \boldsymbol{\Sigma}^{MO_i}\left(\boldsymbol{\Sigma}^{OO_i}\right)^{-1}\boldsymbol{\Sigma}^{OM_i} \tag{3.11}$$

as shown in [4, Thm. 2.5.1]. The conditional means and variances of each missing value are then found by extracting the appropriate element from $\tilde{\boldsymbol{\mu}}_i^{M_i}$ or the diagonal of $\tilde{\boldsymbol{\Sigma}}_i^{MM_i}$. In the context of distance estimation, only the value of the mean and covariance are relevant, and the full distribution is not important.

If the assumption of a normal distribution is not true, the actual distribution must have smaller entropy. Hence Eq. (3.11) will lead to over-estimating the conditional variance, and subsequently to over-estimating the expected distance by Eq. (3.8). In machine learning, over-estimating the distance may be preferable to under-estimating, as this minimises the chance of false positives when looking for nearby samples.

Estimating the covariance matrix

While calculating the mean is simple even when the data has missing values, it is not as easy to determine the best way to estimate the covariance matrix. The two standard approaches are often inadequate:

Available-case analysis in this case refers to separately estimating the covariance for each pair of variables, including every sample for which the two variables are observed. This approach can however result in a matrix which is not positive definite, which leads to further problems when trying to solve a linear system using a part of it (as in Eq. (3.10)).

Complete-case analysis means ignoring all incomplete samples. The usefulness of this approach depends entirely on how many complete samples there are left, and whether that is enough to get a decent estimate.

The most accurate method is usually some variant of the EM algorithm to find a maximum likelihood estimate, even though it is computationally somewhat more demanding.

3.1.3 Using Gaussian mixture models

More accurate estimates for the distances can be derived by fitting a Gaussian mixture model to the data, as studied in Publication II.

Fitting the mixture model using the EM algorithm in Section 2.5.2 provides the conditional means $\tilde{\boldsymbol{x}}_{ik}$ and covariances $\tilde{\boldsymbol{\Sigma}}_{ik}$ for each sample with respect to each mixture component. It only remains to determine the overall conditional mean and covariance matrix. These are found weighted by the memberships as follows:

$$\tilde{\boldsymbol{x}}_i = \sum_{k=1}^{K} t_{ik} \tilde{\boldsymbol{x}}_{ik}, \qquad \tilde{\boldsymbol{\Sigma}}_i = \sum_{k=1}^{K} t_{ik} \left(\tilde{\boldsymbol{\Sigma}}_{ik} + \tilde{\boldsymbol{x}}_{ik} \tilde{\boldsymbol{x}}_{ik}^T \right) - \tilde{\boldsymbol{x}}_i \tilde{\boldsymbol{x}}_i^T. \tag{3.12}$$

The expression for the covariance is found by direct calculation of the second moments. In order to estimate pairwise distances, the conditional variances $\tilde{\sigma}_{i,l} = \tilde{\boldsymbol{\Sigma}}_i^{ll}$ can be extracted from the diagonal of the conditional covariance matrix, or s_i calculated directly as the trace of $\tilde{\boldsymbol{\Sigma}}_i$.

3.1.4 Extension to weighted distances

The same idea can also be used to estimate the Mahalanobis distances, or any such metric weighted by a positive definite matrix $\boldsymbol{S}^{-1}$ written in the form:

$$\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_S^2 = (\boldsymbol{x}_i - \boldsymbol{x}_j)^T \boldsymbol{S}^{-1} (\boldsymbol{x}_i - \boldsymbol{x}_j) \tag{3.13}$$

As the matrix $\boldsymbol{S}$ is positive definite, it's inverse has a Cholesky decomposition $\boldsymbol{S}^{-1} = \boldsymbol{L}\boldsymbol{L}^T$. Then:

$$\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_S^2 = (\boldsymbol{x}_i - \boldsymbol{x}_j)^T \boldsymbol{S}^{-1} (\boldsymbol{x}_i - \boldsymbol{x}_j) = \|\boldsymbol{L}^T \boldsymbol{x}_i - \boldsymbol{L}^T \boldsymbol{x}_j\|^2 = \sum_{l=1}^{d} (L_l^T \boldsymbol{x}_i - L_l^T \boldsymbol{x}_j)^2 \tag{3.14}$$

where L_l is the lth column of $\boldsymbol{L}$. Applying Eq. (3.8):

$$\mathrm{E}\left[\|\boldsymbol{L}^T \boldsymbol{x}_i - \boldsymbol{L}^T \boldsymbol{x}_j\|^2\right] = \|\boldsymbol{L}^T \tilde{\boldsymbol{x}}_i - \boldsymbol{L}^T \tilde{\boldsymbol{x}}_j\|^2 + \sum_{l=1}^{d} \mathrm{Var}[L_l^T \boldsymbol{x}_i] + \sum_{l=1}^{d} \mathrm{Var}[L_l^T \boldsymbol{x}_j] \tag{3.15}$$

Now, using the conditional covariance matrices $\tilde{\boldsymbol{\Sigma}}_i^{MM_i}$ corresponding to each sample $\boldsymbol{x}_i$ and the fact that the variance of a sum is the sum of the covariances [99, corollary 5.4]:

$$\mathrm{Var}[L_l^T \boldsymbol{x}_i] = \mathrm{Var}\left[\sum_{j=1}^{d} L_{jl} x_{i,j}\right] = \sum_{j \in M_i} \sum_{k \in M_i} L_{jl} L_{kl} \,\mathrm{Cov}[X_{i,j}, X_{i,k}] = L_l^T \tilde{\boldsymbol{\Sigma}}_i L_l \tag{3.16}$$

Here $\tilde{\boldsymbol{\Sigma}}_i$ is the conditional covariance matrix corresponding to the sample $\boldsymbol{x}_i$, with zeros for any covariances involving observed values. In terms of the matrix $\tilde{\boldsymbol{\Sigma}}_i^{MM_i}$ from Eq. (3.11), it would be

$$\tilde{\boldsymbol{\Sigma}}_i = \begin{pmatrix} \boldsymbol{0}^{OO_i} & \boldsymbol{0}^{OM_i} \\ \boldsymbol{0}^{MO_i} & \tilde{\boldsymbol{\Sigma}}_i^{MM_i} \end{pmatrix}. \tag{3.17}$$

The relevant part is the sum of these variances, and the sum over the diagonal elements is the trace. Since the trace of a matrix product is invariant under cyclic permutations, this can be written as:

$$\sum_{l=1}^{d} \mathrm{Var}[L_l^T \boldsymbol{x}_i] = \sum_{l=1}^{d} L_l^T \tilde{\boldsymbol{\Sigma}}_i L_l = \mathrm{tr}\left(\boldsymbol{L}^T \tilde{\boldsymbol{\Sigma}}_i \boldsymbol{L}\right) = \mathrm{tr}\left(\boldsymbol{L}\boldsymbol{L}^T \tilde{\boldsymbol{\Sigma}}_i\right) = \mathrm{tr}\left(\boldsymbol{S}^{-1} \tilde{\boldsymbol{\Sigma}}_i\right) \tag{3.18}$$

Putting it all together, the expected squared Mahalonobis distance is straightforward to calculate:

$$\mathrm{E}\left[\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_S^2\right] = \|\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_j\|_S^2 + s_i^2 + s_j^2 \quad \text{where} \quad s_i^2 = \mathrm{tr}\left(\boldsymbol{S}^{-1} \tilde{\boldsymbol{\Sigma}}_i\right). \tag{3.19}$$

This can be seen as a generalisation of Eq. (3.8).

3.1.5 Experiments

To compare the different methods for estimating distances, a simulated experiment is done on the Boston housing data set, whereby values are removed at random. Three separate experiments are done, from a low ratio of missing values (5%) to medium (20%) and high (50%). The accuracy of the estimated distances is then used to compare. The Gaussian mixture model is compared to the model of a single multivariate normal distribution, the partial distance strategy (PDS), and calculating the distances after Incomplete-case k-NN Imputation (ICkNNI).

First, the methods are compared by the root mean squared error (RMSE) of all the estimated pairwise distances in the data set,

$$C_1 = \left(\frac{1}{\kappa}\sum_{i>j}(\hat{d}(\boldsymbol{x}_i,\boldsymbol{x}_j) - d(\boldsymbol{x}_i,\boldsymbol{x}_j))^2\right)^{1/2} \tag{3.20}$$

where $d(\boldsymbol{x}_i,\boldsymbol{x}_j)$ is the true Euclidean distance between samples i and j calculated without any missing data, and $\hat{d}(\boldsymbol{x}_i,\boldsymbol{x}_j)$ is the estimate of the distance provided by each method after removing data. The scaling factor κ is determined so that the average is calculated only over those distances which are estimates, discarding all the cases where the distance can be calculated exactly because neither sample has any missing components: $\kappa = MN - M(M+1)/2$, where M is the number of samples having missing values.

A common application for pairwise distances is a nearest neighbour search, and thus the average (true) distance to the predicted nearest neighbour is used as a second criterion,

$$C_2 = \frac{1}{N}\sum_{i=1}^{N} d(\boldsymbol{x}_i,\boldsymbol{x}_{\mathrm{NN}(i)}), \quad \text{where} \quad \mathrm{NN}(i) = \arg\min_{j\neq i}\hat{d}(\boldsymbol{x}_i,\boldsymbol{x}_j) \tag{3.21}$$

Here, NN(i) is the nearest neighbour of the ith sample as estimated by the method, and $d(\boldsymbol{x}_i,\boldsymbol{x}_{\mathrm{NN}(i)})$ is the true Euclidean distance between the samples as calculated without any missing data. The criterion measures how well the method can identify samples which actually are close in the real data.

The average RMSE values for the methods are presented in Table 3.1. The best result for each row is underlined, and any results which are not statistically significantly different (two-tailed paired t-test, $\alpha = 0.05$) from the best result are bolded. The values in parenthesis represent the accuracy when the distances are calculated using the particular model for imputation only.

Table 3.2 shows the corresponding performances in terms of the true distance to the predicted nearest neighbour.

It can be seen that the mixture model approach generally leads to the best result, while PDS has the largest errors in all cases. The results also clearly show that including the variance terms of Equation (3.8) leads to an improvement in the accuracy compared to only imputing the values. Publications I and II include further experiments on several other data sets, with similar results.

Table 3.1. Average RMSE of estimated pairwise distances, comparing the two proposed variants the Partial Distance Strategy (PDS) and Incomplete-case k-NN Imputation (ICkNNI).

	PDS	ICkNNI	Single Gaussian		Mixture model	
5%	0.514	**0.329**	0.338	(0.348)	**0.331**	(0.338)
20%	1.001	0.672	0.597	(0.650)	**0.587**	(0.619)
50%	2.269	1.593	**1.066**	(1.330)	1.104	(1.245)

Table 3.2. Average of the mean distance to the estimated nearest neighbour, comparing the two proposed variants the Partial Distance Strategy (PDS) and Incomplete-case k-NN Imputation (ICkNNI).

	PDS	ICkNNI	Single Gaussian		Mixture model	
5%	1.047	0.901	0.911	(0.907)	**0.886**	(0.894)
20%	1.790	1.376	1.299	(1.309)	**1.237**	(1.277)
50%	3.692	2.744	**2.086**	(2.228)	**2.073**	(2.225)

3.2 Machine learning using estimated distances

3.2.1 Using estimated distances for a kernel matrix

Several machine learning methods can be formulated in terms of the distances between samples, for instance LS-SVM (Section 2.2.4). The kernel matrix is generally required to be positive semi-definite, and such a kernel is known as a Mercer kernel [51]. Using the distance estimation procedure in Section 3.1 will result in a valid kernel in exactly the same way as a distance matrix calculated on fully observed data.

To see this, it is sufficient to show that the estimated distance matrix is a valid Euclidean distance matrix. This can be done by explicitly constructing a set of points with the required distance matrix in a higher-dimensional ($d+N$-dimensional) space as follows:

- The first d components of each point $\boldsymbol{x}_i$ as per the conditional expectation $\tilde{x}_{i,l}$ in Eq. (3.5).
- Each point $\boldsymbol{x}_i$ is offset by s_i from Eq. (3.8) in a direction orthogonal to everything else

Calculating the squared Euclidean distance between points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in this space exactly leads to Eq. (3.8). As the matrix of estimated pairwise distances is equal to a matrix of pairwise distances (in another space), the kernel matrix will be positive-definite for any appropriate kernel function.

3.2.2 ELM with missing values

Using distance estimation to construct an extreme learning machine is discussed in Publications II and III. It is achieved by selecting the activation function appropriately, so that it can be expressed in terms of distances. The RBF kernel is thus a natural choice. RBF neural networks are commonly trained by choosing the centres by a clustering method or other optimisation procedure [51, 10]. In order to achieve the random projection property, which is essential for ELM, the parameters for the hidden layer nodes should be assigned randomly [54, 55]. The kernel centres $\boldsymbol{\mu}_j$ can be either selected from among the training samples, or as random points in the input space. The widths σ_j can be randomly drawn from an appropriate distribution. The training phase then consists in finding the least-squares solution to the linear system

$$\mathbf{H}\boldsymbol{\beta} = \boldsymbol{y} \tag{3.22}$$

where $\boldsymbol{y}$ is the target output of the labelled data and the hidden layer output matrix $\mathbf{H}$ has the elements

$$H_{ij} = \exp\left(-\frac{\left\|\boldsymbol{x}_i - \boldsymbol{\mu}_j\right\|^2}{\sigma_j^2}\right) \tag{3.23}$$

augmented by a constant column of ones to account for the bias term.

Using any method to estimate the distances between each sample $\boldsymbol{x}_i$ and centre $\boldsymbol{\mu}_j$, the ELM can be applied in the standard way after inserting the estimates directly into Eq. (3.23).

3.2.3 Experiment on regression

Again, the different distance estimation methods can be compared in their use for regression in the Boston housing data set. Table 3.3 shows the average test errors of ELM models which have been built using different distance estimates, but are otherwise identical.

The mixture model and single Gaussian models lead to similar results in terms of accuracy. Both are clearly better than PDS and ICkNNI, particularly for a larger fraction of missing values. Several further experiments with ELM models for both regression and classification tasks are presented in Publication II.

Table 3.3. Average normalised MSE of ELM predictions for regression tasks.

	PDS	ICkNNI	Single Gaussian		Mixture model	
5%	0.242	0.199	0.199	(0.199)	**0.198**	(**0.198**)
20%	0.342	0.279	**0.255**	(0.256)	**0.255**	(0.258)
50%	0.567	0.593	**0.419**	(0.446)	0.433	(0.461)

3.3 Time series modelling with Gaussian mixtures

Gaussian mixture models can also be used as an effective time series model to accomplish prediction and gap-filling, and this method has been introduced in Publication IV. A rolling window is used to extract sub-sequences of length d. The next step is a time-delay embedding, where each sub-sequence is interpreted as a point in $\mathbb{R}^d$. The coordinates are determined by the respective values of the time series.

A Gaussian mixture model can be fit to the data in the d-dimensional space by the EM algorithm, appropriately marginalising over any missing values. Additional constraints are applied to ensure that the covariance structure of the mixture model is consistent with the autoregressive time series configuration.

3.3.1 Fitting the model

Starting with a time series z of length n

$$z_0, z_1, z_2, \ldots, z_{n-2}, z_{n-1},$$

after fixing a regressor length d, conduct a delay embedding [60] by constructing the design matrix $\mathbf{X}$:

$$\mathbf{X} = \begin{bmatrix} z_0 & z_1 & \dots & z_{d-1} \\ z_1 & z_2 & \dots & z_d \\ \vdots & \vdots & & \vdots \\ z_{n-d} & z_{n-d+1} & \dots & z_{n-1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_{n-d} \end{bmatrix}. \tag{3.24}$$

The rows of $\mathbf{X}$ should be seen as vectors in $\mathbb{R}^d$. The idea is to use GMM to estimate the density of these points. The EM algorithm from Section 2.5.1 works well to fit the model.

Gaps in the time series lead to missing values in the design matrix. However, this is not a problem since the EM algorithm can account for missing values (Section 2.5.2).

Missing-data padding

Since the EM algorithm deals with missing values, it makes sense to construct $\mathbf{X}$ by considering that everything before and after the measurement period consists of "missing values". This is called *padding* the design matrix $\mathbf{X}$ with missing values (marked as '?'). The procedure maximises the use of the data for training, and effectively increases the number of available training samples from $n-d+1$ to $n+d-1$ (cf. Eq. (3.24)):

$$\mathbf{X} = \begin{bmatrix} ? & ? & \dots & ? & z_0 \\ ? & ? & \dots & z_0 & z_1 \\ \vdots & \vdots & & \vdots & \vdots \\ ? & z_0 & \dots & z_{d-3} & z_{d-2} \\ z_0 & z_1 & \dots & z_{d-2} & z_{d-1} \\ \vdots & \vdots & & \vdots & \vdots \\ z_{n-d} & z_{n-d+1} & \dots & z_{n-2} & z_{n-1} \\ z_{n-d+1} & z_{n-d+2} & \dots & z_{n-1} & ? \\ \vdots & \vdots & & \vdots & \vdots \\ z_{n-1} & ? & \dots & ? & ? \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_{d-2} \\ \boldsymbol{x}_{d-1} \\ \vdots \\ \boldsymbol{x}_{n-1} \\ \boldsymbol{x}_n \\ \vdots \\ \boldsymbol{x}_{n+d-2} \end{bmatrix} \tag{3.25}$$

Another advantage is that any available-data analysis over a subset of variables is invariant to shifts in the indices, as it should be for time series.

3.3.2 Constrained covariance model

The GMM is intended for estimating arbitrary continuous distributions, and thus ignores some issues specific to time series. In particular, the

mean of a stationary time series is the same no matter what lag you are observing it at, so the mean of every variable in the GMM should also be the same. Also, the $d \times d$ covariance matrix corresponding to the GMM distribution represents the autocovariance matrix of the time series up to lag $d-1$, and should thus be a symmetric Toeplitz matrix. A Toeplitz matrix is a matrix which is constant along every diagonal.

There are various ways to create a mixture model which satisfies the constraints, but the best approach is to incorporate the restrictions into the EM algorithm while fitting the model to data.

Having means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$, and mixing coefficients π_k for each component k of a GMM, these parameters can be used to calculate the mean and covariance of the full distribution:

$$\boldsymbol{\mu} = \sum_{k=1}^{K} \pi_k \boldsymbol{\mu}_k, \qquad \boldsymbol{\Sigma} = \sum_{k=1}^{K} \pi_k \left(\boldsymbol{\Sigma}_k + \boldsymbol{\mu}_k \boldsymbol{\mu}_k^T \right) - \boldsymbol{\mu}\boldsymbol{\mu}^T. \tag{3.26}$$

Now the mean should be made equal for each variable, and the matrix should be made Toeplitz and symmetric:

$$\boldsymbol{\Sigma} \approx \mathbf{R}_z = \begin{bmatrix} r_z(0) & r_z(1) & r_z(2) & \dots & r_z(d-1) \\ r_z(1) & r_z(0) & r_z(1) & \dots & r_z(d-2) \\ \vdots & \vdots & \vdots & & \vdots \\ r_z(d-1) & r_z(d-2) & r_z(d-3) & \dots & r_z(0) \end{bmatrix} \tag{3.27}$$

$\mathbf{R}_z$ is the autocovariance matrix of the time series z, with $r_z(l)$ being the autocovariance at lag l.

The idea here is to first go through a standard iteration of the EM algorithm. After calculating the parameters in the M-step, they should be modified by as little as possible so that the constraints are satisfied.

Let $\boldsymbol{\theta} = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k\}_{k=1}^K$ be a structure containing the current parameters of the GMM, and Ω the space of all such models. Then define $\Phi \subset \Omega$ as the subset of parameter sets which satisfy the constraints

$$\Phi = \{\boldsymbol{\theta} \in \Omega \mid \boldsymbol{\mu} \text{ is equal and } \boldsymbol{\Sigma} \text{ is Toeplitz}\} \tag{3.28}$$

When maximising the expected log-likelihood with the constraints, the M-step should be

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta} \in \Phi} Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}), \tag{3.29}$$

but this is not feasible to solve exactly. Instead, first calculate the standard M-step

$$\boldsymbol{\theta}' = \arg\max_{\boldsymbol{\theta} \in \Omega} Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}), \tag{3.30}$$

and then project the result $\boldsymbol{\theta}'$ onto Φ to find the closest solution

$$\boldsymbol{\theta}^{(t+1)} = \arg\min_{\boldsymbol{\theta}\in\Phi} d(\boldsymbol{\theta}, \boldsymbol{\theta}') \tag{3.31}$$

for an appropriate interpretation of the distance $d(\boldsymbol{\theta}, \boldsymbol{\theta}')$ between models. The intuition is that if the difference between models is minimised, their log-likelihoods should not be too far off. Specifically, $Q(\boldsymbol{\theta}^{(t+1)} \mid \boldsymbol{\theta}^{(t)})$ should be close to the optimal $\max_{\boldsymbol{\theta}\in\Phi} Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)})$.

As the quantity is not maximised, even though it can be observed to increase, this is strictly not an EM algorithm. Instead, it is an instance of the *Generalised EM* (GEM) algorithm. As long as an increase is ensured in every iteration, the GEM algorithm converges similarly as the EM algorithm [24, 74].

To achieve the result in practice, define the distance function between sets of parameters as follows:

$$d(\boldsymbol{\theta}, \boldsymbol{\theta}') = \sum_{k=1}^{K} \left\| \boldsymbol{\mu}_k - \boldsymbol{\mu}'_k \right\|^2 + \sum_{k=1}^{K} \left\| \boldsymbol{S}_k - \boldsymbol{S}'_k \right\|_F^2 + \sum_{k=1}^{K} (\pi_k - \pi'_k)^2 , \tag{3.32}$$

where $\boldsymbol{S}_k = \boldsymbol{\Sigma}_k + \boldsymbol{\mu}_k \boldsymbol{\mu}_k^T$ are the second moments of the distributions of each component and $\|\cdot\|_F$ is the Frobenius norm. Using Lagrange multipliers, it can be shown that this distance function is minimised by the results presented below in Eqs. (3.34) and (3.37).

The mean

After an iteration of the normal EM-algorithm by Eqs. (2.38–2.40), find the vector with equal components which is nearest to the global mean $\boldsymbol{\mu}$ as calculated by Eq. (3.26). This is done by finding the mean m of the components of $\boldsymbol{\mu}$, and calculating the discrepancy $\boldsymbol{\delta}$ of how much the current mean is off from the equal mean:

$$m = \frac{1}{d} \sum_{j=1}^{d} \mu_j , \qquad \boldsymbol{\delta} = \boldsymbol{\mu} - m\mathbf{1} , \tag{3.33}$$

where $\mathbf{1}$ is a vector of ones. Shift the means of each component to compensate, as follows:

$$\boldsymbol{\mu}'_k = \boldsymbol{\mu}_k - \frac{\pi_k}{\sum_{j=1}^{K} \pi_j^2} \boldsymbol{\delta} \qquad \forall k . \tag{3.34}$$

As can be seen, components with larger π_k take on more of the "responsibility" of the discrepancy, as they contribute more to the global statistics. Any weights which sum to unity would fulfil the constraints, but choosing the weights to be directly proportional to π_k minimises the distance in Eq. (3.32).

The covariance

After updating the means $\boldsymbol{\mu}_k$, recalculate the covariances around the updated values as

$$\hat{\boldsymbol{\Sigma}}_k = \boldsymbol{\Sigma}_k + \boldsymbol{\mu}_k \boldsymbol{\mu}_k^T - \boldsymbol{\mu}_k' \boldsymbol{\mu}_k'^T \qquad \forall k\,. \tag{3.35}$$

The global covariance from Eq. (3.26))

$$\hat{\boldsymbol{\Sigma}} = \sum_{k=1}^{K} \pi_k \left(\hat{\boldsymbol{\Sigma}}_k + \boldsymbol{\mu}_k' \boldsymbol{\mu}_k'^T \right) - \boldsymbol{\mu}' \boldsymbol{\mu}'^T\,. \tag{3.36}$$

Then, find the nearest (in Frobenius norm) Toeplitz matrix $\mathbf{R}$ by calculating the mean of each diagonal of $\hat{\boldsymbol{\Sigma}}$:

$$r(0) = \frac{1}{d} \sum_{j=1}^{d} \hat{\Sigma}_{j,j}, \qquad r(1) = \frac{1}{d-1} \sum_{j=1}^{d-1} \hat{\Sigma}_{j,j+1}, \qquad r(2) = \frac{1}{d-2} \sum_{j=1}^{d-2} \hat{\Sigma}_{j,j+2}, \qquad \text{etc.}$$

The discrepancy $\boldsymbol{\Delta}$ from this Toeplitz matrix is

$$\boldsymbol{\Delta} = \hat{\boldsymbol{\Sigma}} - \mathbf{R}, \quad \text{where } \mathbf{R} = \begin{bmatrix} r(0) & r(1) & r(2) & \dots & r(d-1) \\ r(1) & r(0) & r(1) & \dots & r(d-2) \\ \vdots & \vdots & \vdots & & \vdots \\ r(d-1) & r(d-2) & r(d-3) & \dots & r(0) \end{bmatrix}.$$

In order to satisfy the constraint of a Toeplitz matrix for the global covariance, the component covariances are updated as

$$\boldsymbol{\Sigma}_k' = \hat{\boldsymbol{\Sigma}}_k - \frac{\pi_k}{\sum_{j=1}^{K} \pi_j^2} \boldsymbol{\Delta} \qquad \forall k\,, \tag{3.37}$$

the weights being the same as in Eq. (3.34). Eqs. (3.34) and (3.37), together with $\pi_k' = \pi_k$, minimise the distance in Eq. (3.32) subject to the constraints.

Heuristic correction

Unfortunately, the procedure described above does not account for the spectral composition of the matrices, and can occasionally lead to matrices $\boldsymbol{\Sigma}_k'$ which are not positive definite. Hence, an additional heuristic correction c_k is applied in such cases to force the matrix to remain positive definite:

$$\boldsymbol{\Sigma}_k'' = \hat{\boldsymbol{\Sigma}}_k - \frac{\pi_k}{\sum_{k=1}^{K} \pi_k^2} \boldsymbol{\Delta} + c_k \mathbf{I} \qquad \forall k. \tag{3.38}$$

In the experiments section of Publication IV, the value $c_k = 1.1|\lambda_{k0}|$ is used, where λ_{k0} is the most negative eigenvalue of $\boldsymbol{\Sigma}_k'$. The multiplier needs to be larger than unity to avoid making the matrix singular.

A more appealing correction would be to only increase the negative (or zero) eigenvalues to some acceptable, positive, value. However, this would

break the constraint of a Toeplitz global covariance matrix, and hence the correction must be applied to all eigenvalues, as is done in Eq. (3.38) by adding to the diagonal.

Free parameters

The constraints reduce the number of free parameters relevant to calculating the AIC and BIC. Without constraints, the number of free parameters is

$$P = \underbrace{Kd}_{\text{means}} + \underbrace{\frac{1}{2}Kd(d+1)}_{\text{covariances}} + \underbrace{K-1}_{\text{mixing coeffs}} , \tag{3.39}$$

where K is the number of Gaussian components, and d is the regressor length. There are $d-1$ equality constraints for the mean, and $\frac{1}{2}d(d-1)$ constraints for the covariance, each reducing the number of free parameters by 1. With the constraints, the number of free parameters is then

$$P' = \underbrace{(K-1)d+1}_{\text{means}} + \underbrace{\frac{1}{2}(K-1)d(d+1)+d}_{\text{covariances}} + \underbrace{K-1}_{\text{mixing coeffs}} . \tag{3.40}$$

The leading term is reduced from $\frac{1}{2}Kd^2$ to $\frac{1}{2}(K-1)d^2$, in effect allowing one additional component for approximately the same number of free parameters.

Exogenous time series or non-contiguous lag

If the design matrix is formed in a different way than by taking consecutive values, the restrictions for the covariance matrix will change. Such cases are handled by forcing any affected elements in the matrix to equal the mean of the elements it should equal. This will also affect the number of free parameters.

As this sort of delay embedding may inherently have a low intrinsic dimension, optimising the selection of variables could considerably improve the accuracy of the model.

3.3.3 Forecasting and gap-filling

The model readily lends itself to being used for short-to-medium term time series prediction. For example, if a time series is measured monthly and displays some seasonal behaviour, a Gaussian model could be trained with a regressor size of 24 (two years). This allows us to take the last year's measurements as the 12 *first* months, and determine the conditional expectation of the following 12 months.

The mixture model provides a direct way to calculate the conditional expectation. Let the input dimensions be partitioned into past values P (known) and future values F (unknown). Then, given a sample $\boldsymbol{x}_i^P$ for which only the past values are known and a prediction is to be made, calculate the probabilities of it belonging to each component

$$t_{ik} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_i^P \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_i^P \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}, \tag{3.41}$$

where $\mathcal{N}(\boldsymbol{x}_i^P \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is the *marginal* multivariate normal distribution probability density of the observed (i.e., past) values of $\boldsymbol{x}_i$.

Let the means and covariances of each component also be partitioned according to past and future variables:

$$\boldsymbol{\mu}_k = \begin{bmatrix} \boldsymbol{\mu}_k^P \\ \boldsymbol{\mu}_k^F \end{bmatrix}, \quad \boldsymbol{\Sigma}_k = \begin{bmatrix} \boldsymbol{\Sigma}_k^{PP} & \boldsymbol{\Sigma}_k^{PF} \\ \boldsymbol{\Sigma}_k^{FP} & \boldsymbol{\Sigma}_k^{FF} \end{bmatrix}. \tag{3.42}$$

Then the conditional expectation of the future values with respect to the component k is given by

$$\tilde{\boldsymbol{z}}_{ik} = \boldsymbol{\mu}_k^F + \boldsymbol{\Sigma}_k^{FP} (\boldsymbol{\Sigma}_k^{PP})^{-1} (\boldsymbol{x}_i^P - \boldsymbol{\mu}_k^P) \tag{3.43}$$

The total conditional expectation can now be found as a weighted average of these predictions by the probabilities t_{ik}:

$$\hat{\boldsymbol{z}}_i = \sum_{k=1}^{K} t_{ik} \tilde{\boldsymbol{y}}_{ik}. \tag{3.44}$$

It should be noted that the method directly estimates the full vector of future values at once, in contrast with most other methods which would separately predict each required data point.

To conduct missing value imputation, the procedure is the same as for prediction. The only difference is that in this case the index set P contains all known values for a sample (both before and after the target to be predicted), while F contains the missing values that will be imputed.

3.3.4 Experiment

To illustrate the use for modelling gapped time series, some experimental results are shown here. The studied time series is the Santa Fe time series competition data set A: Laser generated data [110]. The task is set at predicting the next 12 values, given the previous 12. This makes the regressor size $d = 24$, and the mixture model fitting is in a 24-dimensional space.

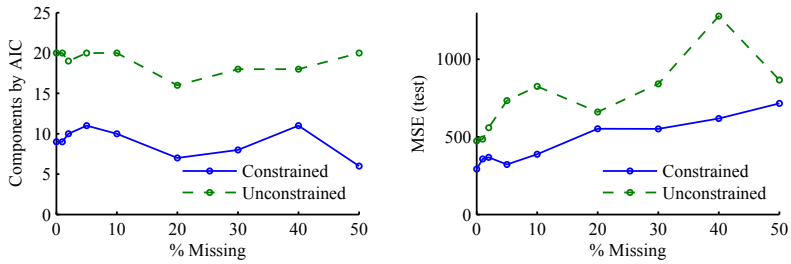

Figure 3.1. Results on the Santa Fe A Laser time series data for various degrees of missing values, including the number of components selected by AIC and the resulting MSEs of the corresponding test set predictions.

The modelling is repeated with various degrees of missing data (1% through 50%). In the training phase, missing data is removed at random from the time series before forming the padded design matrix. To calculate the testing MSE, missing values are also removed from the inputs (i.e., the past values from which predictions are to be made) at the same probability. The MSE is then calculated as the error between the forecast and the actual time series (with no values removed). The entire procedure is done twice – with and without the constraints in section 3.3.2 – to show how the constraints affect the results.

Fig. 3.1 shows the number of components selected by AIC (Section 2.5.3), and the corresponding test MSEs, for various degrees of missing values. As expected, the forecasting accuracy deteriorates with an increasing ratio of missing data. The number of components selected by the AIC remains largely constant, and the constrained model consistently performs better. Publication IV includes further details concerning the experiment.

4. Variable Selection Methods

4.1 Mutual information estimation by Gaussian mixtures

The mutual information between input and output variables is a natural choice for a variable selection criterion, and several methods for estimating it have been proposed (see Section 2.3.2). Publication V presents how a mixture of Gaussians can be used for this purpose. There are several reasons which make Gaussian mixtures an appealing method for estimating mutual information for feature selection:

1. After fitting the mixture model to the full set of variables, the model can directly be used to calculate the mutual information for any subset of variables. This is useful in variable selection, where it is often necessary to evaluate a large number of different subsets.

2. Estimates for different variable sets seem to behave more consistently than with using other estimators. In particular, the estimate of the mutual information nearly always increases when adding variables, as it should.

3. As the Gaussian mixture can be fit to data with missing values, the estimator works for such incomplete data sets as well.

Mutual information estimators directly based on estimating the probability density of the underlying probability distribution of the data have generally been discouraged in the literature due to the difficulty of obtaining accurate estimates of the density. However, as Publication V shows, Gaussian mixture models can be used very effectively for this purpose.

The main idea is to use a Gaussian mixture model to estimate the densities of the variables. However, instead of directly calculating Eq. (2.20), consider Eq. (2.19), and interpret the integral as an expectation.

$$I(X;Y) = \int_Y \int_X p(x,y) \log\left(\frac{p(x,y)}{p_x(x)p_y(y)}\right) dx\,dy \tag{4.1}$$

$$= \mathrm{E}\left[\log p(x,y) - \log p_x(x) - \log p_y(y)\right] \tag{4.2}$$

Given a sample of data $\{x_i, y_i\}_{i=1}^N$, the expectation can be approximated by the arithmetic mean over the data:

$$\hat{I}(X;Y) = \frac{1}{N}\sum_{i=1}^{N}\left(\log p(x_i,y_i) - \log p_x(x_i) - \log p_y(y_i)\right) \tag{4.3}$$

The proposed approach is based on this expression, requiring only estimates of the density and marginal density at each point of data. By fitting a Gaussian mixture model to the joint space $X \times Y$, the resulting model directly provides an estimate of $p(x_i, y_i)$. To calculate the marginal probability densities, the *same* Gaussian model is used, restricted to the appropriate variables. The marginal model is easily acquired by only including the appropriate elements from the means and covariances of each Gaussian component. Having a GMM with K components in the $X \times Y$ space with mixing coefficients π_k, means $\boldsymbol{\mu}_k$, and covariances $\boldsymbol{\Sigma}_k$ for each component k ($0 < \pi_k < 1$, $\sum_{k=1}^K \pi_k = 1$), the parameters can be partitioned as below:

$$\boldsymbol{\mu}_k = \begin{bmatrix} \boldsymbol{\mu}_k^X \\ \boldsymbol{\mu}_k^Y \end{bmatrix}, \quad \boldsymbol{\Sigma}_k = \begin{bmatrix} \boldsymbol{\Sigma}_k^{XX} & \boldsymbol{\Sigma}_k^{XY} \\ \boldsymbol{\Sigma}_k^{YX} & \boldsymbol{\Sigma}_k^{YY} \end{bmatrix}. \tag{4.4}$$

The marginal model for X is directly determined as a GMM of K components with the same mixing coefficients π_k, but means $\boldsymbol{\mu}_k^X$ and covariance matrices $\boldsymbol{\Sigma}_k^{XX}$. The marginal GMM is similarly found for Y, and for any subspaces of X corresponding to different sets of selected variables.

As the goal is to evaluate differences between the joint density $p(x,y)$ and the product $p_x(x)p_y(y)$, the same model should be used to estimate all the quantities. It might seem reasonable to separately optimise another mixture model in the space for X to estimate p_x instead, and this could result in a more accurate estimate for p_x itself, but could also lead to spurious differences causing an inflated KL divergence. Having consistent estimates is particularly important for variable selection, where mutual information estimates for different variable sets are compared to each other.

In machine learning, the goal is to find a model that can predict an output variable Y from several input variables X, and here the mutual information with the output is used to select the variables. However, the mutual information never decreases when adding irrelevant variables. Thus an exhaustive search over all feature sets is meaningless, as it is known beforehand that the criterion is maximised when all the variables are included. The forward search is a more practical approach; here variables are added one by one, at each step selecting the variable which leads to the largest increase in MI when considered together with the previously selected variables. The order of successive selection then leads to a ranking of variables: the first selected variable can be seen as the most important, and so on.

4.2 The Delta test

The Delta test is the leave-one-out error of the nearest neighbour (1-NN) regression model. While the model itself is not particularly accurate compared to more sophisticated regression models, it can be used for variable selection by choosing those variables which minimise the error.

The method was initially studied in Publication VI, and it has been used with success in several cases [82, 8, 95, 71, 112, 70, 35, 113]. Publication VII presents further theoretical justification to explain why the method works as well as it does. Several papers specifically focused on optimising the Delta test have also been published in the literature [44, 72, 73, 97, 96, 43, 45].

4.2.1 Noise variance estimation

The Delta test is traditionally considered a method for residual noise variance estimation. In the kind of regression tasks considered here, the data consist of N input points $\{\boldsymbol{x}_i\}_{i=1}^N$ and associated scalar outputs $\{y_i\}_{i=1}^N$ [58]. The assumption is that there is a functional dependence between them with an additive noise term:

$$y_i = f(\boldsymbol{x}_i) + \varepsilon_i \tag{4.5}$$

The function f is often assumed to be smooth, or at least continuous, and the additive noise terms ε_i are i.i.d. with zero mean and finite variance. Noise variance estimation is the study of how to find an a priori estimate for $\mathrm{Var}(\varepsilon)$ given some data without considering any specifics of the shape

of f. Having a reliable estimate of the amount of noise is useful for model structure selection and determining when a model may be overfitting.

The original formulation [81] of the Delta test was based on the concept of variable-sized neighbourhoods, but an alternative formulation [98] with a first-nearest-neighbour (NN) approach has later surfaced. In this treatment, specifically this 1-NN formulation will be used as there is no parameter to select, and it is conceptually and computationally simple. The Delta test could also be seen as an extension of the Rice variance estimator [86] to multivariate data.

The nearest neighbour of a point is defined as the unique point in a data set which minimises a distance metric to that point:

$$\mathrm{NN}(i) := \arg\min_{j \neq i} \left\| \boldsymbol{x}_i - \boldsymbol{x}_j \right\|^2. \tag{4.6}$$

It may occur that the nearest neighbour is not unique, and in that case it is sufficient to randomly pick one from the set of nearest neighbours. In this context, the neighbours are determined by the Euclidean distance. It may be justified to use other metrics to get better results in some cases, if some input variables are known to have specific characteristics that the Euclidean metric fails to account for appropriately. Knowing if other metrics are more appropriate generally requires external knowledge about the source or behaviour of the data. For instance, data representing class labels are best handled by the discrete metric, and "time-of-day" or "time-of-year"-type variables by taking into account their cyclic behaviour.

The Delta test, initially introduced in [81] and further developed in [98], is usually written as

$$\delta = \frac{1}{2N} \sum_{i=1}^{N} \left(y_i - y_{\mathrm{NN}(i)} \right)^2, \tag{4.7}$$

i.e., the differences in the outputs associated with neighbouring (in the input space) points are considered. This is a well-known estimator of $\mathrm{Var}(\varepsilon)$ and it has been shown—e.g., in [68]—that the estimate converges to the true value of the noise variance in the limit $N \to \infty$. Although it is not considered to be the most accurate noise estimator, its advantages include reliability, simplicity, and computational efficiency [58]. The method appears not to be particularly sensitive to mild violations of the assumptions made about the data, such as independence and distributions of the noise terms.

4.2.2 The Delta test for variable selection

The Delta test was originally intended to be used for estimating residual variance. Following [95, 112], Publications VI and VII examine a different use: to use it as a cost function for variable selection by choosing that selection of variables which minimises the Delta test. Each subset of variables can be mapped to a value of the estimator by evaluating the expression in Eq. (4.7) so that the nearest neighbours NN(i) are determined according to the distance in the subspace spanned by the subset of variables. Define the Delta test $\delta : \mathscr{P}(I) \to \mathbb{R}$ as

$$\delta(\tilde{I}) := \frac{1}{2N} \sum_{i=1}^{N} \left(y_i - y_{\mathrm{NN}(i;\tilde{I})} \right)^2 \tag{4.8}$$

where

$$\mathrm{NN}(i;\tilde{I}) := \arg\min_{j \neq i} \left\| \boldsymbol{x}_i - \boldsymbol{x}_j \right\|_{\tilde{I}}^2, \tag{4.9}$$

and the distances are calculated considering only the current set of variables:

$$\left\| \boldsymbol{x}_i - \boldsymbol{x}_j \right\|_{\tilde{I}}^2 := \sum_{k \in \tilde{I}} \left(x_{i,k} - x_{j,k} \right)^2. \tag{4.10}$$

The publications show that choosing the subset which gives the smallest value for the Delta test constitutes an effective variable selection procedure for regression modelling. As the purpose of the Delta test is to deal with noisy data, it is impossible to formulate a statement showing that the Delta test could always choose the perfect variables, due to the random effects of the noise. Hence the assertions consider the expectation of the Delta test, and show that the expectation is minimised for the best selection of variables for a finite number N of data points.

In Publication VII, it is shown that with a finite (but sufficiently large) number of samples, the expectation of the Delta test is uniquely minimised by the smallest selection of variables which can fully explain the deterministic part of the target function. Combined with the property that the variance of the Delta test converges to zero with increasing number of samples, this suggests that the probability of getting the correct selection generally increases with the amount of data available.

Some assumptions concerning the distribution of the data are required in order for the results to hold true. These continuity assumptions detailed in Publication VII are designed to be similar to and compatible with the assumptions many popular non-linear modelling techniques make about the data. This enhances the usability of the Delta test as a preprocessing step for practically any non-linear regression task.

An exhaustive search over the $2^d - 1$ non-empty subsets of d variables to find the global optimum of the Delta test is a possibility, but not feasible for large values of d. Instead, there are more efficient approximate search schemes. Stepwise search methods, where variables are individually added or removed depending on the change in the criterion are often practical. Search strategies for variable selection are further discussed in detail in, e.g., [47, 85].

Publications VI and VII only consider using the Delta test for regression problems, but the method could also be applied to classification tasks by considering the misclassification rate of the nearest neighbour classifier. Its use for classification has not been extensively tested, however, and the analysis in Publication VII only applies to regression tasks.

4.3 Experiments

The two variable selection methods presented – mutual information estimation by Gaussian mixtures and the Delta test – are here experimentally evaluated against four other variable selection/ranking methods:

1. LARS: Least angle regression [28] (Section 2.3.1)

2. RReliefF [88], the regression variant of the Relief method (Section 2.3.3)

3. Mutual Information by Kraskov's estimator [63]

4. Variable selection by Maximum Likelihood Mutual Information (MLMI) estimation [101, 102]

The comparison criterion is the mean squared error of a least squares support vector machine (LS-SVM) [100] regression model, as this model is known to be sensitive to redundant variables. The model is trained using the selected variable set, and the median (over repeated runs of optimising hyperparameters) leave-one-out error is calculated. This can be considered a fair criterion for comparing the *selections* of variables. As a preprocessing step, all variables including the target variable are standardised to zero mean and unit variance before the variable selection process.

The Delta test is optimised by an exhaustive search over all possible

selections. The other criteria are used with a forward search approach for selecting variables. This results in a ranking of variables, and the variable sets formed by successively selecting the selected variables are evaluated by the resulting LS-SVM accuracy.

For the Boston housing data set, the modelling task is to predict the value of a house/apartment from the 13 other properties. The variables selected by the methods as well as the median LOO-errors of the LS-SVM are all presented in Table 4.1. There are no obviously redundant variables in the data set, as is evidenced by the constantly decreasing error when successively choosing the variables determined by each ranking method. The only exception is found by Kraskov's estimator, which manages to find a better performing set of variables by excluding 2, 4, and 11. The Delta test find the exactly same result, the final selection including all but those three variables.

The same test is also applied to forecasting the Santa Fe A time series. It has been shown that a regressor size of 12 should suffice to train an efficient model. The variable selection then pertains to which of the delayed regressors (up to a delay of 12) should be used to build the model. The results are shown in Table 4.2. The best accuracy is obtained by choosing the top three variables as ranked by RReliefF. The Delta test performs decently, leading to a better model than the Kraskov and MLMI estimators, while choosing only three of the regressor variables. The Gaussian mixture here outperforms the Delta test, as well as the other MI estimators.

More experiments comparing the variable selection methods on several additional data sets are presented in Publications V and VII.

Table 4.1. The selected inputs and median LOO MSE for the Boston housing data. Bold values represent optimal choices in the sense of the lowest error with the smallest set of variables.

	1	2	3	4	5	6	7	8	9	10	11	12	13	MSE
DT	•		•		•	•	•	•	•	•		•	•	**0.0892**

LARS		RReliefF		Kraskov		MLMI		GMM	
13	**0.3236**	6	0.4257	13	**0.3236**	13	**0.3236**	13	**0.3236**
6	**0.2323**	13	**0.2323**	6	**0.2323**	11	0.2416	6	**0.2323**
11	0.2037	5	0.1901	10	**0.1516**	5	0.2018	11	0.2037
12	0.1909	8	0.1739	5	**0.1476**	10	0.1918	8	0.1531
4	0.1772	4	0.1829	9	**0.1305**	9	0.1946	5	0.1359
1	0.1544	10	0.1571	1	**0.1158**	3	0.1893	4	0.1434
8	0.1435	12	0.1379	12	0.1205	6	0.1167	12	0.1366
5	0.1331	2	0.1316	7	0.1176	8	**0.1129**	2	0.1395
2	0.1360	9	0.1150	8	**0.0964**	7	0.1094	1	0.1360
3	0.1290	11	0.1067	3	**0.0892**	4	0.1148	9	0.1237
9	0.1161	3	0.1054	4	0.0991	12	0.0956	10	0.1062
10	0.1048	7	0.0953	2	0.0982	2	0.0953	3	0.1048
7	0.0926	1	0.0926	11	0.0926	1	0.0926	7	0.0926

Table 4.2. The selected inputs and median LOO MSE for the Santa Fe A data. Bold values represent optimal choices in the sense of the lowest error with the smallest set of variables.

	1	2	3	4	5	6	7	8	9	10	11	12	MSE
DT	•	•										•	0.0143

LARS		RReliefF		Kraskov		MLMI		GMM	
8	**0.3750**	1	0.6537	8	**0.3750**	4	0.4643	7	0.4224
7	0.1250	2	**0.0208**	7	0.1250	2	0.0839	1	0.1770
3	0.1044	8	**0.0087**	1	0.0811	6	0.0728	2	0.0136
1	0.0203	9	0.0144	9	0.0718	8	0.0650	6	0.0142
2	0.0137	7	0.0147	6	0.0735	5	0.0619	5	0.0159
4	0.0138	10	0.0243	3	0.0175	7	0.0643	10	0.0309
5	0.0144	6	0.0228	5	0.0172	3	0.0627	3	0.0315
6	0.0152	3	0.0179	4	0.0178	10	0.0724	4	0.0328
9	0.0156	11	0.0219	2	0.0156	12	0.0885	8	0.0177
10	0.0206	12	0.0257	10	0.0206	1	0.0187	9	0.0206
12	0.0220	5	0.0241	11	0.0221	11	0.0193	11	0.0221
11	0.0245	4	0.0245	12	0.0245	9	0.0245	12	0.0245

5. Conclusion

This dissertation considers two separate aspects of machine learning that are particularly relevant when taking standard methods and using them for practical tasks.

Distance estimation is introduced as a fresh approach to machine learning with missing values. It is shown that finding the expected value of the squared distance between two samples reduces to calculating the conditional mean and variance of each missing value separately. Conducting the estimation using a Gaussian mixture or a single Gaussian are both shown to be effective procedures for different use cases. Using the estimated distance for further machine learning methods leads to more accurate models than simply filling in the missing values.

For time series modelling, the mixture of Gaussians also proves to be very useful when appropriate restrictions on the model are applied, especially considering its inherent ability to deal with gaps in the data.

Two different methods for subset evaluation are studied for automated variable selection. The Delta test is a simple criterion which is fast to calculate, but is nevertheless powerful at identifying relevant variables from redundant ones. Using Gaussian mixtures to estimate mutual information is another particularly effective method to differentiate between variables. While fitting the mixture can be computationally expensive, the same model can subsequently be used to evaluate all the required variable subsets.

The work on this dissertation commenced with studying two separate issues, but in the end introduces a method to address the combined problem of variable selection *with* incomplete data, by mutual information estimation with Gaussian mixtures.

While the methods presented show promising results in several applications, some limitations must be recognised. The procedures rely on the as-

sumption of having a mixture of Gaussians which approximates the probability density with reasonable accuracy. Fitting the mixture, however, can be difficult to accomplish well, as it involves many case-dependent choices. The general model with fully independent covariance matrices for each component is not always the most practical; instead, diagonal matrices, shared covariances, or other restrictions can be appropriate. Determining the number of components is also not straightforward, even though several criteria have been developed to select this value. Even then, the EM algorithm may converge slowly, or not at all, and end up in local optima. These issues combined make the process not only computationally expensive, but burdensome for the end user.

A limitation of the time series modelling approach is that it requires the original data to have been evenly sampled with equal spacing. This assumption is shared with all other autoregressive models, but does not hold for many practical applications where time series are gathered.

The variable selection methods define a criteria for evaluating a selection of variables, but do not provide any assistance in how to search the space of possible subsets. For high-dimensional applications where variable selection is most needed, an exhaustive search is not tractable. Several methods do exist to facilitate the search, but finding a good search strategy is not a trivial issue.

Certain issues related to the methods presented in this thesis have not been fully resolved as of yet. These open questions include the following:

- Which accuracy criterion for estimated distances is most relevant in machine learning with missing data?

- Which model selection criterion for determining the number of components in a Gaussian mixture leads to the best results in the type of problems studied here?

- Delta test: can it be proven that the probability of getting the correct (i.e., minimal fully explaining) selection of variables converges to 1 with increasing N?

- Both variable selection methods presented only work with regression tasks. What is the best way to extend them to (potentially multi-class) classification problems?

These questions, and the ongoing research on other methods, show that there remains work to be done in this problem area. Data gathering technology has reached a point where data is a available to such an extent that appropriate procedures for analysis and interpretation are still catching up. As such, there is some way to go before reaching a future with more automated machine learning systems to help the users extract maximally meaningful information from ever-increasing data collections.

Bibliography

[1] Hirotugu Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, December 1974.

[2] Ethem Alpaydin. *Introduction to Machine Learning*. MIT Press, 2004.

[3] Edoardo Amaldi and Viggo Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209:237–260, 1998.

[4] T. W. Anderson. *An Introduction to Multivariate Statistical Analysis*. Wiley-Interscience, third edition, 2003.

[5] Kevin Bache and Moshe Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

[6] Roberto Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5(4): 537–550, July 1994.

[7] Richard E. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, Princeton, New Jersey, U.S.A., 1961.

[8] Souhaib Ben Taieb, Antti Sorjamaa, and Gianluca Bontempi. Multiple-output modeling for multi-step-ahead time series forecasting. *Neurocomputing*, 73(10–12):1950–1957, 2010.

[9] Caroline Beunckens, Geert Molenberghs, and Michael G. Kenward. Direct likelihood analysis versus simple forms of imputation for missing data in randomized clinical trials. *Clinical Trials*, 2(5):379–386, 2005.

[10] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1996.

[11] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[12] Åke Björck. *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics, 1996.

[13] Charles Bouveyron, Stéphane Girard, and Cordelia Schmid. High-dimensional data clustering. *Computational Statistics & Data Analysis*, 52(1):502–519, 2007.

[14] George E. P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. *Time Series Analysis: Forecasting and Control*. Wiley Series in Probability and Statistics. John Wiley & Sons, fourth edition, 2008.

[15] Raymond B. Cattell. The scree test for the number of factors. *Multivariate Behavioral Research*, 1(2):245–276, 1966.

[16] Gavin C. Cawley and Nicola L. C. Talbot. Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. *Neural Networks*, 17(10):1467–1475, 2004.

[17] Chun-houh Chen, Wolfgang Härdle, Antony Unwin, Michael A. A. Cox, and Trevor F. Cox. Multidimensional scaling. In *Handbook of Data Visualization*, Springer Handbooks of Computational Statistics, pages 315–347. Springer Berlin Heidelberg, 2008.

[18] Jiahua Chen and Jun Shao. Nearest neighbor imputation for survey data. *Journal of Official Statistics*, 16(2):113–131, 2000.

[19] John G. Cleary and Leonard E. Trigg. K*: An instance-based learner using an entropic distance measure. In *12th International Conference on Machine Learning*, pages 108–114, 1995.

[20] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[21] Marie Cottrell and Patrick Letrémy. Missing values: processing with the Kohonen algorithm. In *Proc. International Symposium on Applied Stochastic Models and Data Analysis (ASMDA)*, pages 489–496, 2005.

[22] Thomas M. Cover. Estimation by the nearest neighbor rule. *IEEE Transactions on Information Theory*, 14(1):50–55, January 1968.

[23] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, second edition, 2006.

[24] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):pp. 1–38, 1977.

[25] John K. Dixon. Pattern recognition with partly missing data. *IEEE Transactions on Systems, Man and Cybernetics*, 9(10):617–621, October 1979.

[26] Gauthier Doquire and Michel Verleysen. Feature selection with missing data using mutual information estimators. *Neurocomputing*, 90:3–11, 2012.

[27] Ke-Lin Du and M. N. S. Swamy. *Neural Networks and Statistical Learning*. Springer-Verlag London, 2014.

[28] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

[29] Craig K. Enders. *Applied Missing Data Analysis*. Methodology In The Social Sciences. Guilford Press, 2010.

[30] Alireza Farhangfar, Lukasz Kurgan, and Jennifer Dy. Impact of imputation of missing values on classification error for discrete data. *Pattern Recognition*, 41(12):3692–3705, 2008.

[31] Damien François. *High-dimensional data analysis: from optimal metrics to feature selection*. VDM Verlag Dr. Muller, 2008.

[32] Damien François, Fabrice Rossi, Vincent Wertz, and Michel Verleysen. Resampling methods for parameter-free and robust feature selection with mutual information. *Neurocomputing*, 70(7–9):1276–1288, 2007.

[33] Benoît Frénay, Gauthier Doquire, and Michel Verleysen. Is mutual information adequate for feature selection in regression? *Neural Networks*, 48: 1–7, 2013.

[34] Benoît Frénay, Gauthier Doquire, and Michel Verleysen. Theoretical and empirical study on the potential inadequacy of mutual information for feature selection in classification. *Neurocomputing*, 112:64–78, 2013.

[35] Raimundo Garcia-del Moral, Alberto Guillén, Luis Javier Herrera, Antonio Cañas, and Ignacio Rojas. Parametric and non-parametric feature selection for kidney transplants. In Ignacio Rojas, Gonzalo Joya, and Joan Cabestany, editors, *Advances in Computational Intelligence*, volume 7903 of *Lecture Notes in Computer Science*, pages 72–79. Springer Berlin Heidelberg, 2013.

[36] Pedro J. García-Laencina, José-Luis Sancho-Gómez, Aníbal R. Figueiras-Vidal, and Michel Verleysen. K nearest neighbours with mutual information for simultaneous classification and missing data imputation. *Neurocomputing*, 72(7–9):1483–1493, 2009.

[37] Zoubin Ghahramani and Michael I. Jordan. Learning from incomplete data. Technical report, Lab Memo No. 1509, CBCL Paper No. 108, MIT AI Lab, 1995.

[38] Andrej Gisbrecht, Dušan Sovilj, Barbara Hammer, and Amaury Lendasse. Relevance learning for time series inspection. In *ESANN 2012 Proceedings – 20th European Symposium on Artificial Neural Networks*, pages 489–494, 2012.

[39] Andrej Gisbrecht, Yoan Miche, Barbara Hammer, and Amaury Lendasse. Visualizing dependencies of spectral features using mutual information. In *ESANN 2013 Proceedings – 21st European Symposium on Artificial Neural Networks*, pages 573–578, 2013.

[40] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, third edition, 1996.

[41] Jerzy W. Grzymala-Busse and Witold J. Grzymala-Busse. Handling missing attribute values. In Oded Maimon and Lior Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pages 33–51. Springer US, second edition, 2010.

[42] Peter D. Grünwald. *The Minimum Description Length Principle*. MIT Press, 2007.

[43] Alberto Guillén, Mark van Heeswijk, Dušan Sovilj, Maribel García Arenas, Luis Javier Herrera, Hector Pomares, and Ignacio Rojas. Variable selection in a GPU cluster using Delta test. In *IWANN (1)*, pages 393–400, 2011.

[44] Alberto Guillén, Dušan Sovilj, Fernando Mateo, Ignacio Rojas, and Amaury Lendasse. Minimizing the Delta test for variable selection in regression problems. *International Journal of High Performance Systems Architecture*, 1(4):269–281, 2008.

[45] Alberto Guillén, M. Isabel García Arenas, Mark van Heeswijk, Dušan Sovilj, Amaury Lendasse, Luis Javier Herrera, Héctor Pomares, and Ignacio Rojas. Fast feature selection in a GPU cluster using the Delta test. *Entropy*, 16(2):854–869, 2014.

[46] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[47] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lotfi A. Zadeh, editors. *Feature Extraction: Foundations and Applications*. Springer, 2006.

[48] Mark A. Hall. *Correlation-based Feature Selection for Machine Learning*. PhD thesis, University of Waikato, April 1999.

[49] Fredric M. Ham and Ivica Kostanic. *Principles of Neurocomputing for Science and Engineering*. McGraw-Hill, 2001.

[50] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[51] Simon Haykin. *Neural Networks and Learning Machines*. Pearson Education, third edition, 2009.

[52] Ludmila Himmelspach and Stefan Conrad. Clustering approaches for data with missing values: Comparison and evaluation. In *International Conference on Digital Information Management (ICDIM)*, pages 19–28, July 2010.

[53] Eduardo R. Hruschka, Estevam R. Hruschka Jr., and Nelson F. F. Ebecken. Evaluating a nearest-neighbor method to substitute continuous missing values. In *AI 2003: Advances in Artificial Intelligence*, volume 2903 of *Lecture Notes in Computer Science*, pages 723–734. Springer Berlin Heidelberg, 2003.

[54] Guang-Bin Huang and Chee-Kheong Siew. Extreme learning machine with randomly assigned RBF kernels. *International Journal of Information Technology*, 11(1):16–24, 2005.

[55] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1–3):489–501, 2006.

[56] Lynette Hunt and Murray Jorgensen. Mixture model clustering for mixed data with missing information. *Computational Statistics & Data Analysis*, 41(3–4):429–440, 2003.

[57] Clifford M. Hurvich and Chih-Ling Tsai. Regression and time series model selection in small samples. *Biometrika*, 76(2):297–307, 1989.

[58] Antonia J. Jones. New tools in non-linear modelling and prediction. *Computational Management Science*, 1(2):109–149, 2004.

[59] Per Jönsson and Claes Wohlin. An evaluation of k-nearest neighbour imputation using Likert data. In *Proc. International Symposium on Software Metrics*, pages 108–118, September 2004.

[60] Holger Kantz and Thomas Schreiber. *Nonlinear Time Series Analysis*. Cambridge nonlinear science series. Cambridge University Press, second edition, 2003.

[61] Kenji Kira and Larry A. Rendell. A practical approach to feature selection. In *Proceedings of the ninth international workshop on Machine learning*, ML92, pages 249–256. Morgan Kaufmann, 1992.

[62] Teuvo Kohonen. *Self-Organizing Maps*. Springer Series in Information Sciences. Springer Berlin Heidelberg, 2001.

[63] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Phys. Rev. E*, 69:066138, June 2004.

[64] Nojun Kwak. Feature extraction based on direct calculation of mutual information. *International Journal of Pattern Recognition and Artificial Intelligence*, 21(07):1213–1231, 2007.

[65] Nojun Kwak and Chong-Ho Choi. Input feature selection by mutual information based on Parzen window. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1667–1671, 2002.

[66] John A. Lee and Michel Verleysen. *Nonlinear Dimensionality Reduction*. Springer, 2007.

[67] Jiye Li and N. Cercone. Assigning missing attribute values based on rough sets theory. In *IEEE International Conference on Granular Computing*, pages 607–610. IEEE Computer Society, May 2006.

[68] Elia Liitiäinen, Michel Verleysen, Francesco Corona, and Amaury Lendasse. Residual variance estimation in machine learning. *Neurocomputing*, 72(16–18):3692–3703, 2009.

[69] Roderick J. A. Little and Donald B. Rubin. *Statistical Analysis with Missing Data*. Wiley-Interscience, second edition, 2002.

[70] F. Liébana-Cabanillas, R. Nogueras, L. J. Herrera, and A. Guillén. Analysing user trust in electronic banking using data mining methods. *Expert Systems with Applications*, 40(14):5439–5447, 2013.

[71] Fernando Mateo and Amaury Lendasse. A variable selection approach based on the Delta test for extreme learning machine models. In *Proceedings of the European Symposium on Time Series Prediction*, pages 57–66, September 2008.

[72] Fernando Mateo, Dušan Sovilj, Rafael Gadea, and Amaury Lendasse. RCGA-S/RCGA-SP methods to minimize the Delta test for regression tasks. In *Bio-Inspired Systems: Computational and Ambient Intelligence*, volume 5517 of *Lecture Notes in Computer Science*, pages 359–366. Springer, 2009.

[73] Fernando Mateo, Dušan Sovilj, and Rafael Gadea. Approximate k-NN Delta test minimization method using genetic algorithms: Application to time series. *Neurocomputing*, 73(10-12):2017–2029, 2010.

[74] Geoffrey McLachlan and Thriyambakam Krishnan. *The EM Algorithm and Extensions*. Wiley Series in Probability and Statistics. John Wiley & Sons, New York, 1997.

[75] Geoffrey J. McLachlan and David Peel. *Finite Mixture Models*. Wiley Series in Probability and Statistics. John Wiley & Sons, New York, 2000.

[76] Yoan Miche, Antti Sorjamaa, Patrick Bas, Olli Simula, Christian Jutten, and Amaury Lendasse. OP-ELM: Optimally-pruned extreme learning machine. *IEEE Transactions on Neural Networks*, 21(1):158–162, 2010.

[77] Yoan Miche, Mark van Heeswijk, Patrick Bas, Olli Simula, and Amaury Lendasse. TROP-ELM: a double-regularized ELM using LARS and Tikhonov regularization. *Neurocomputing*, 74(16):2413–2421, 2011.

[78] Andrew C. Morris, Martin P. Cooke, and Phil D. Green. Some solutions to the missing feature problem in data classification, with application to noise robust ASR. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 737–740, May 1998.

[79] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[80] Amir Navot, Lavi Shpigelman, Naftali Tishby, and Eilon Vaadia. Nearest neighbor based feature selection for regression and its application to neural activity. In *Advances in Neural Information Processing Systems 18*, pages 995–1002. MIT Press, Cambridge, MA, 2006.

[81] Hong Pi and Carsten Peterson. Finding the embedding dimension and variable dependencies in time series. *Neural Computation*, 6(3):509–520, 1994.

[82] Federico Montesino Pouzols and Angel Barriga Barros. Automatic clustering-based identification of autoregressive fuzzy inference models for time series. *Neurocomputing*, 73(10–12):1937–1949, 2010.

[83] Joaquin Quiñonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence, editors. *Dataset Shift in Machine Learning*. MIT Press, 2009.

[84] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[85] Juha Reunanen. *Overfitting in Feature Selection: Pitfalls and Solutions*. PhD thesis, Aalto University, 2012.

[86] John Rice. Bandwidth choice for nonparametric regression. *The Annals of Statistics*, 12(4):1215–1230, 1984.

[87] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14 (5):465–471, 1978.

[88] Marko Robnik-Šikonja and Igor Kononenko. Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning*, 53(1-2):23–69, October 2003.

[89] Fabrice Rossi. Model collisions in the dissimilarity SOM. In *ESANN 2007 Proceedings – 15th European Symposium on Artificial Neural Networks*, pages 25–30, 2007.

[90] Fabrice Rossi, Amaury Lendasse, Damien François, Vincent Wertz, and Michel Verleysen. Mutual information for the selection of relevant variables in spectrometric nonlinear modelling. *Chemometrics and Intelligent Laboratory Systems*, 80(2):215–226, 2006.

[91] Donald B. Rubin. *Multiple Imputation for Nonresponse in Surveys*. Wiley Series in Probability and Statistics. Wiley, 1987.

[92] Joseph L. Schafer and John W. Graham. Missing data: Our view of the state of the art. *Psychological Methods*, 7(2):147–177, 2002.

[93] Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[94] Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk, editors. *Nearest-Neighbor Methods in Learning and Vision*. MIT Press, 2006.

[95] Antti Sorjamaa, Jin Hao, Nima Reyhani, Yongnan Ji, and Amaury Lendasse. Methodology for long-term prediction of time series. *Neurocomputing*, 70(16–18):2861–2869, October 2007.

[96] Dušan Sovilj. Multistart strategy using Delta test for variable selection. In Timo Honkela, Włodzisław Duch, Mark Girolami, and Samuel Kaski, editors, *Artificial Neural Networks and Machine Learning – ICANN 2011, Proceedings, Part II*, volume 6792 of *Lecture Notes in Computer Science*, pages 413–420. Springer, 2011.

[97] Dušan Sovilj, Antti Sorjamaa, and Yoan Miche. Tabu search with Delta test for time series prediction using OP-KNN. In *Proceedings of the European Symposium on Time Series Prediction*, pages 187–196, September 2008.

[98] Aðalbjörn Stefánsson, Nenad Končar, and Antonia J. Jones. A note on the gamma test. *Neural Computing & Applications*, 5(3):131–133, 1997.

[99] Charles J. Stone. *A Course in Probability and Statistics*. Duxbury Press, 1995.

[100] Johan A. K. Suykens, Tony Van Gestel, Jos De Brabanter, Bart De Moor, and Joos Vandewalle. *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002.

[101] Taiji Suzuki, Masashi Sugiyama, Jun Sese, and Takafumi Kanamori. Approximating mutual information by maximum likelihood density ratio estimation. *JMLR Workshop and Conference Proceedings*, 4:5–20, 2008.

[102] Taiji Suzuki, Masashi Sugiyama, and Toshiyuki Tanaka. Mutual information approximation via maximum likelihood estimation of density ratio. In *IEEE International Symposium on Information Theory (ISIT 2009)*, pages 463–467, 2009.

[103] Hiroshi Tenmoto, Mineichi Kudo, and Masaru Shimbo. Mdl-based selection of the number of components in mixture models for pattern classification. In Adnan Amin, Dov Dori, Pavel Pudil, and Herbert Freeman, editors, *Advances in Pattern Recognition*, volume 1451 of *Lecture Notes in Computer Science*, pages 831–836. Springer Berlin Heidelberg, 1998.

[104] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*, 58:267–288, 1996.

[105] Volker Tresp, Subutai Ahmad, and Ralph Neuneier. Training neural networks with deficient data. In *Advances in Neural Information Processing Systems 6*, pages 128–135. Morgan Kaufmann, 1994.

[106] Jason Van Hulse and Taghi M. Khoshgoftaar. Incomplete-case nearest neighbor imputation in software measurement data. *Information Sciences*, 259:596–610, 2014.

[107] Michel Verleysen, Damien François, Geoffroy Simon, and Vincent Wertz. On the effects of dimensionality on data analysis with neural networks. In *Artificial Neural Nets Problem solving methods*, Lecture Notes in Computer Science 2687, pages II105–II112. Springer-Verlag, 2003.

[108] Zs. J. Viharos, L. Monostori, and T. Vincze. Training and application of artificial neural networks with incomplete data. In Tim Hendtlass and Moonis Ali, editors, *Developments in Applied Artificial Intelligence*, volume 2358 of *Lecture Notes in Computer Science*, pages 649–659. Springer Berlin Heidelberg, 2002.

[109] Ito Wasito and Boris Mirkin. Nearest neighbour approach in the least-squares data imputation algorithms. *Information Sciences*, 169(1-2):1–25, 2005.

[110] Andreas S. Weigend and Neil A. Gershenfeld. The Santa Fe time series competition data, 1991. URL `http://www-psych.stanford.edu/~andreas/Time-Series/SantaFe.html`.

[111] Andreas S. Weigend and Neil A. Gershenfeld, editors. *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison-Wesley, Reading, MA, 1994.

[112] Qi Yu, Eric Séverin, and Amaury Lendasse. A global methodology for variable selection: Application to financial modeling. In *Mashs 2007, Computational Methods for Modelling and learning in Social and Human Sciences, Brest (France)*, May 2007.

[113] Qi Yu, Mark van Heeswijk, Yoan Miche, Rui Nian, Bo He, Eric Séverin, and Amaury Lendasse. Ensemble Delta test-extreme learning machine (DT-ELM) for regression. *Neurocomputing*, 129:153–158, 2014.

DISSERTATIONS IN INFORMATION AND COMPUTER SCIENCE

Aalto-DD205/2013 Lijffijt, Jefrey
Computational methods for comparison and exploration of event sequences. 2013.

Aalto-DD21/2014 Cho, Kyunghyun
Foundations and Advances in Deep Learning. 2014.

Aalto-DD49/2014 Lindh-Knuutila, Tiina
Computational Modeling and Simulation of Language and Meaning: Similarity-Based Approaches. 2014.

Aalto-DD80/2014 Toivola, Janne
Advances in Wireless Damage Detection for Structural Health Monitoring. 2014.

Aalto-DD105/2014 Parkkinen, Juuso
Probabilistic components of molecular interactions and drug responses. 2014.

Aalto-DD108/2014 Faisal, Ali
Retrieval of Gene Expression Measurements with Probabilistic Models. 2014.

Aalto-DD110/2014 Virtanen, Seppo
Bayesian latent variable models for learning dependencies between multiple data sources. 2014.

Aalto-DD120/2014 Bergström-Lehtovirta, Joanna
The Effects of Mobility on Mobile Input. 2014.

Aalto-DD127/2014 Zhang, He
Advances in Nonnegative Matrix Decomposition with Application to Cluster Analysis. 2014.

Aalto-DD138/2014 Sovilj, Dušan
Learning Methods for Variable Selection and Time Series Prediction. 2014.

Aalto-DD 144/2014

ISBN 978-952-60-5870-2
ISBN 978-952-60-5871-9 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934
ISSN 1799-4942 (pdf)

Aalto University
School of Science
Department of Information and Computer Science
www.aalto.fi

BUSINESS +
ECONOMY

ART +
DESIGN +
ARCHITECTURE

SCIENCE +
TECHNOLOGY

CROSSOVER

**DOCTORAL
DISSERTATIONS**