Learning Methods for Variable Selection and Time Series Prediction

Dušan Sovilj

Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 138/2014

Learning Methods for Variable Selection and Time Series Prediction

Dušan Sovilj

A doctoral dissertation completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Science, at a public examination held at the lecture hall T2 of the school on 31 October 2014 at 12 noon.

Aalto University
School of Science
Department of Information and Computer Science

Supervising professor
Prof. Juha Karhunen

Thesis advisors
Dr. Amaury Lendasse
Dr. Federico Montesino Pouzols

Preliminary examiners
Prof. Michel Verleysen, Université catholique de Louvain, Belgium
Dr. Klaus Neumann, Bielefeld University, Germany

Opponent
Prof. Tommi Kärkkäinen, University of Jyväskylä, Finland

Aalto University publication series
DOCTORAL DISSERTATIONS 138/2014

ISBN 978-952-60-5856-6
ISBN 978-952-60-5857-3 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-60-5857-3

Unigrafia Oy
Helsinki 2014

Finland

441 697
Printed matter

Aalto University

Abstract

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

Author
Dušan Sovilj

Name of the doctoral dissertation
Learning Methods for Variable Selection and Time Series Prediction

Publisher School of Science

Unit Department of Information and Computer Science

Series Aalto University publication series DOCTORAL DISSERTATIONS 138/2014

Field of research Information and Computer Science

Manuscript submitted 9 June 2014 **Date of the defence** 31 October 2014

Permission to publish granted (date) 28 August 2014 **Language** English

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

Abstract

In the recent years, machine learning methods have become increasingly popular for modelling many different phenomena: financial markets, spatio-temporal data sets, pattern recognition, speech and image processing, recommender systems and many others. This huge interest in machine learning comes from the great success of their application and the increasingly easier acquisition, storage and access of data.

In this thesis, two general problems in machine learning are discussed and several solutions are offered. The first problem is variable selection, an approach to automatically select the most relevant features in the data. Two key phases of variable selection are the search criterion and the search algorithm. The thesis focuses on the Delta test as a search criterion, while several solutions are offered for the search algorithm, such as the Genetic Algorithm and Tabu Search. Furthermore, the selection procedure is extended for more general cases of scaling and projection, as well as their combination. Finally, some of the above proposed solutions have been developed for parallel architectures which enable the whole variable selection procedure to be used for data sets with a high number of features.

The second problem tackled in the thesis is time series prediction that arises in many fields of science and industry. In simple words: time series prediction involves the estimation of future values for a series of measurements of a/the phenomenon of interest. The number of these estimations can be small, leading to short-term prediction, or several hundreds which constitute long-term prediction. Two models have been developed for this particular task. One is based on a recently popular neural network type called Extreme Learning Machine, while the other is a juxtaposition of Generative Topographic Mapping and Relevance Learning modified for regression tasks.

Finally, the above problems are tackled together for real-world time series coming from a biological domain. The difficulty of making any kind of inference in biological time series is due to really small amount of available samples, irregular sampling frequency and spatial coverage of areas of interest. Nevertheless, more stable model parameter estimation is possible with the combined use of global climate indicators and regional measurements in the form of a multifactor approach.

Keywords variable selection/scaling/projection, time series prediction, environmental modelling, model structure selection

ISBN (printed) 978-952-60-5856-6 **ISBN (pdf)** 978-952-60-5857-3

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

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

Pages 218 **urn** http://urn.fi/URN:ISBN:978-952-60-5857-3

Preface

This dissertation presents the research I have carried out at the Department of Information and Computer Science, Aalto University School of Science. My doctoral studies, conference trips and research visits could not have been possible without the support of Helsinki Graduate School in Science and Engineering (HECSE). I would also like to acknowledge KAUTE Säätiö for their financial support to finalise my doctoral thesis and Kaj-Mikael Björk at Arcada University of Applied Sciences for giving me the opportunity to continue research before the defence.

I wish to thank my former supervisor Professor Olli Simula for his patience and never-ending support and also my current supervisor Professor Juha Karhunen for providing useful advice in the final stages of my doctoral studies. Big thanks and enormous gratitude go to my instructor and a dear friend Amaury Momo Lendasse who has accepted me in his Time Series Prediction and Chemoinformatics group which is now Environmental and Industrial Machine Learning (EIML) group. Momo is a person who is always by your side no matter what and it was a pleasure and fortune I had such a great instructor who is always finding something positive in every situation. Federico Montesino Pouzols is thanked for providing wonderful discussions about modelling environmental data during his short time as my second instructor.

I am thankful to the pre-examiners of the thesis, Professor Michel Verleysen and Dr. Klaus Neumann for their valuable comments which substantially improved the final manuscript. Furthermore, I am grateful to Professor Tommi Kärkkäinen who kindly accepted to be the opponent for the occasion.

I wish to thank current members of the EIML group: Alexander Grigorievskiy, Anton Akusok, Emil Eirola, Yoan Miche, Francesco Corona and Luiza Sayfullina; and also former members: Antti Sorjamaa, Yu Qi, Zhu

Zhanxing, Yao Li, and Elia Liitiäinen for sharing their stories and providing an enjoyable working environment. Colleague, flatmate and a friend Mark van Heeswijk has been a great company for the past five years, always willing to help and with a positive attitude in encountering any challenge.

I am grateful to Dr. Joachim Dippner for giving me the opportunity to work at the Leibniz Institute for Baltic Sea Research in Rostock and for wonderful hospitality during both of my research visits. Both Joachim and Karin Junker contributed to fruitful discussions about modelling, marine biology and climate phenomena.

My life outside of academia during these long seven years in Finland could not be possible without good friends who provided invaluable company, discussions and entertainment. It is my pleasure to have crossed your paths: Bahram Dastmalchi, Ritabrata Dutta, Kyunghyun Cho, Igor Mataić, Eric Halbach, Yu Bin, Paula Pekkarinen, Karoliina Kekko, Mari-Sanna Paukkeri, André Schumacher, Shinnosuke Seki, Cathy Nangini, Sanja Šćepanović, Magnus Westerlund, Chen Xi and Pekka Kuusela (in no particular order).

I am happy to have parents who have always provided me with endless love and support for all my endeavours.

Finally, my deepest thanks and respect go to Jenni, an amazing lady who has captivated my heart with her wonderful mind and energy.

Helsinki, September 12, 2014,

Dušan Sovilj

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 Dušan Sovilj, Antti Sorjamaa, Qi Yu, Yoan Miche, Eric Séverin. OP-ELM and OP-KNN in Long-Term Prediction of Time Series using Projected Input Data. *Neurocomputing*, 73(10–12):1976–1986, June 2010.

II Fernando Mateo, Dušan Sovilj, Rafael Gadea. Approximate k-NN Delta Test Minimization Method using Genetic Algorithms: Application to Time Series. *Neurocomputing*, 73(10–12):2017–2029, June 2010.

III Karin Junker, Dušan Sovilj, Ingrid Kröncke, Joachim Dippner. Climate induced changes in benthic macrofauna – A non-linear model approach. *Journal of Marine Systems*, 96–97:90–94, August 2012.

IV Dušan Sovilj. Multistart Strategy Using Delta Test for Variable Selection. In *International Conference on Artificial Neural Networks (ICANN 2011, Part II)*, pages 413–420, Lecture Notes in Computer Science volume 6792. Espoo, Finland, June 2011.

V Andrej Gisbrecht, Dušan Sovilj, Barbara Hammer, and Amaury Lendasse. Relevance learning for time series inspection. In *European Symposium on Artificial Neural Networks (ESANN 2012)*, pages 489–494, Computational Intelligence and Machine Learning. Bruges, Belgium, April 2012.

VI Dušan Sovilj, Amaury Lendasse, Olli Simula. Extending Extreme

Learning Machine with Combination Layer. In *International Work-Conference on Artificial Neural Networks*, pages 417—426, Lecture Notes in Computer Science volume 7902. Tenerife, Spain, June 2013.

VII Alberto Guillén, Mark van Heeswijk, Dušan Sovilj, M. G. Arenas, Héctor Pomares, and Ignacio Rojas. Variable Selection in a GPU Cluster using Delta Test. In *International Work-Conference on Artificial Neural Networks*, pages 393–400, Lecture Notes in Computer Science volume 6691. Málaga, Spain, June 2011.

VIII Alberto Guillén, Dušan Sovilj, Mark van Heeswijk, Luis Javier Herrera, Amaury Lendasse, Héctor Pomares, and Ignacio Rojas. Evolutive Approaches for Variable Selection Using a Non-parametric Noise Estimator. *Parallel Architectures & Bioinspired Algorithms*, Studies in Computational Intelligence volume 415, pages 243–266, August 2012.

Notation

$\mathbf{x}_i$	i-th sample in the data set
$\mathbf{X}$	input data matrix
$\mathbf{Y}$	output target vector
f	modelling function
ϵ	additive noise term
σ^2	variance of the noise
N	number of data points
d	dimensionality of data (number of variables)
$NN(i)$	nearest neighbour of sample $\mathbf{x}_i$ in data space
X^j	variable j in the data
S	subset of variables *and*
	a solution to an optimisation problem
I^j	indicator variable for variable X^j
$Ne(\mathbf{I})$	neighbouring solutions of $\mathbf{I}$
F	objective or cost function in optimisation
w^i	scaling weight for variable X^j
H	set of discrete scaling weights
$\mathbf{X}^{\mathrm{P}}$	projected data
$\mathbf{X}^{\mathrm{S}}$	scaled data
$\mathbf{P}$	projection matrix
z_t	time series measurement at time step t
T	length of the time series
h	number of time lags
P	number of parameters in AR model
Q	number of parameters in MA model
$\boldsymbol{\beta}$	parameters of a feed-forward neural network
g_m	m-th neuron in the layer *and*
	an activation function for the same neuron

$\mathbf{T}$	latent space
$\mathcal{M}_i$	i-th model in a pool of solutions
M_i	number of neurons in the i-th hidden layer
M	number of neurons for a network with one hidden layer
$F(\cdot;\boldsymbol{\beta})$	output of a network with parameters $\boldsymbol{\beta}$
κ	number of adjustable parameters for a model
$\mathbf{W}$	parameters of GTM model
K	number of prototypes in GTM
$\Delta(\cdot,\cdot)$	distance function between two samples

Acronyms

AIC	Akaike's Information Criterion
AO	Arctic Oscillation
CV	cross-validation
DT	Delta test
ELM	Extreme Learning Machine
FBS	Forward-Backward Search
GA	Genetic Algorithm
GPU	Graphics Processing Unit
GTM	Generative Topographic Mapping
JMA	Jackknife Model Averaging
LOO	Leave-one-out
MLP	Multi-layer perceptron
NAO	North Atlantic Oscillation
MO	Multi-Objective (optimisation)
OPELM	Optimally Pruned Extreme Learning Machine
OPKNN	Optimally Pruned k-Nearest Neighbour
PCA	Principal Component Analysis
RL	Relevance learning
SOM	Self-Organising Map
TS	Tabu Search

1. Introduction

Machine learning [1–4] has become a popular field of research due to now established and widely successful applications. Any problem involving data gathering and acquisition can benefit from the methods developed in the machine learning community, but also the ones coming from statistics. The interest in these methods has grown over the past few years due to the success of deep learning techniques [5–7] which have now established new state-of-the-art results. Most of the deep learning methods deal with large amounts of data in difficult tasks such as image processing (particularly face recognition), speech processing and video retrieval.

Briefly, machine learning can be described as a way of "learning information" from the available data, where information has somewhat ambiguous notion and can mean many different things which leads to various types of algorithms: clustering [8], feature extraction [9], pattern recognition [1], dimensionality reduction [10], prediction of new samples not used during the training stage, recommender systems [11], factor analysis [12], and many others.

The need for machine learning methods stems from the recent technological advances that enable easier and quicker gathering, storage, transport and access of data. Now it is possible to capture information on a very short time frames since storage has become quite cheap, while small scale monitoring stations and sensors are ubiquitous. The prime example may be the World Wide Web, where millions of people interact and where it is possible to gather the flow of the exchange on a time scale of milliseconds leading to enormously large data sets.

1.1 Scope of the Dissertation

The thesis is concerned with two problems in the machine learning domain. One of the problems is variable selection, an approach to automatically determine relevant features for the data set at hand. The other task is that of time series prediction, in which the prediction for future events is solely based on the past observations. Some of the proposed solutions are applied to biological time series: 1) investigating the connections with climate factors; and 2) empirically validating variable selection methods. These aspects are further discussed in the following paragraphs.

Variable selection. Available data come in the form of samples and features. Features, also known as variables, predictors or inputs, are "independent" measurements that have direct influence on the variable of interest (also called target or dependent variable). The usual assumption in machine learning and statistics is that this connection can be represented as a function, either of known or unknown form. The former case arises when enough prior information about the process is known before the modelling phase, while the latter case is the more common one as most of the time little to no prior knowledge is available. Without any other information besides the data, one cannot reasonably assume any functional form beforehand. In the latter case, one way to study the data is to use clustering techniques where the data points are separated into smaller subregions, or to use, for example, neural networks for modelling any kind of non-pathological (well-behaved containing some structure) data.

Most machine learning algorithms assume that *all* input variables have direct influence on the target variable. However, in certain scenarios this assumption cannot be defended, for example in a situation where such dependence is not known in advance, while the input variables are simply collected because it is easy to measure them. This scenario often arises when the generating process is complex and there is much uncertainty about exact causes and relationships between all involved variables. A simple example is trying to predict a person's shoe size based on their height measurement, while information about their hair colour does not contribute to the actual goal. Such variables (hair colour in this case) are deemed irrelevant or unimportant, and the task of variable selection is to find such features. The problem can be stated in the other direction as well – finding variables which are most relevant for the target variable. Moreover, this issue is even more significant in small sample data

sets where making any kind of inference is quite a challenge. We use the phrase "small sample data" to indicate a data set with small number of samples and both phrases are used interchangeably throughout the thesis. Further complications arise when the number of features is large which is the case in most data sets being collected today. For example, the gene expression data has more than one thousand gene indicators (or features) with only couple of tens of samples. The sheer number of all possible subset combinations is exponential in the number of features and the conventional search techniques are inappropriate for such data sets.

Time series prediction. Time series modelling can be described as having a sequence of measurements coming from a particular source, where the goal is to build a *model* that will enable *prediction* for future instances or values coming from the same source. The sampling frequency between two subsequent measurements is usually assumed to be the same leading to regularly spaced data samples. This research domain has important practical applications as many types of problems can be classified into time series domain, such as financial markets, stock exchange fluctuations, many physical phenomena such as global temperature, number of sunspots, successive Earth's revolutions around the Sun to name a few. The key aspect here is "a model", an object able to capture information from the data and subsequently provide practitioners with the possible outcomes. For example, a retail salesman might be interested in the income for the next day, next two days, or maybe even several weeks in advance, but using only information of the past income periods up to the present time. This example shows that there are two possible goals in time series prediction: short-term and long-term predictions. Short-term approach refers to a situation where one is interested in predicting only a few future values, while long-term means having a model(s) predicting several tens and up to several hundreds of future values of the time series.

Besides prediction, time series can be also analysed for trends, variations, fluctuations, stationarity, interannual cycles and this area is still growing. The task of analysis does not involve any future values which makes it more attractive and less volatile than prediction, but only enables inference about the time period from which the data originates.

Marine biology. Observations for local marine lifeforms only started from 1950's while for some locations it is only a few years old if not couple of months. The main reason behind this is financial, with researchers struggling to find proper support to monitor their local ecosystems. In recent

years, this trend has shifted and many more research institutes are receiving financial support, but the community has recognised the importance of preserving local habitats. Within the European Union there are several projects which aim at monitoring and maintaining marine systems, such as Assessment and Modelling of Baltic Ecosystem Response (AMBER) project[1]. The purpose of monitoring marine species is maintaining the quality of ecosystems, especially their biodiversity. This provides the ecosystems with higher chances of resisting any unusual circumstances, such as abnormally high temperatures, high concentrations of nutrients, low levels of oxygen or dangers coming from the polluted waters.

The fluctuations in species abundance is tightly coupled with both local and global climate changes, and a lot of research is done trying to establish relations between the biological time series with both physical measurements (temperature, salinity, nutrients level) and the climate indicators. Machine learning methods provide a way to analyse these relations and enable a glimpse into the future scenarios about the variability of marine ecosystems. This is basically a time series prediction problem, but at the same time a variable selection problem, and having a suitable model gives the opportunity to enforce preventive measures if any negative effects are predicted.

1.2 Scientific Contributions of the Dissertation

The dissertation contains the following scientific contributions:

- Variable or feature selection with different approaches using a specific criterion, namely the Delta test. Delta test is a noise variance estimator which provides the practitioner with an approximation as to how small or large an error one can expect before doing any modelling on the data. It is based on the nearest neighbours search and is a special case of the Gamma test. As for the different goals, three thematic ones are selection, scaling and projection. Selection basically investigates whether a feature should be included in the set of important features or not. Scaling extends selection to *weight* all the features according to their relevance, while projection deals with altered, that is, projected, data in a new feature space. All of the publications are based on the Delta test as

[1] `www.io-warnemuende.de/amber.html`

the main search criterion, while contributions contain different search algorithms and different approaches to selection (including scaling and projection). These contributions are summarised below:

- Combining scaling and projection into a single projection method (Publication I).

- Developing multistart strategy that does not require any advanced search algorithm besides the basic greedy descent method (Publication IV).

- Employment of the approximate nearest neighbour search to speed up the computations for data sets with large number of features (Publication II).

- Parallel implementation incorporating both global level search (with Genetic Algorithm) and local level refinement (with Tabu Search) for fast feature selection (Publication VIII).

- Implementation on a heterogeneous cluster of computers using all available processing power (both CPU and GPU) with an island model of the Genetic Algorithm (Publication VII).

- Two enhancements to the Extreme Learning Machine (ELM) which is a special type of a feedforward neural network. One contribution deals with the small number of data samples where the basic and first variants of the ELM have problems of building the "optimal" model. Optimality here refers to the appropriate number of neurons in the hidden layer to capture all information in the data. Publication III proposes to use a specific criterion – corrected Akaike's Information Criterion – developed in statistics for the small number of samples. This model is used for predicting the biomass, abundance and number of species for benthic macrofauna in the North Sea. The other contribution (Publication VI) is also tightly related to the model structure selection procedures and proposes a model averaging solution for the hidden layer instead of choosing a single structure from a pool of possibilities. This modification shows a substantial improvement in the prediction task.

- A method incorporating both the feature selection and time series inspection, where the selection is deemed as relevance for the task of prediction. This method (Publication V) extends Generative Topographic Mapping (GTM) and builds an additional layer in which features are weighted for the task at hand. The work is based on the Relevance Learning (RL) techniques and is adapted for the regression task, while the RL method is used for classification tasks. The method enables identification of important time lags, and at the same time provides long-term predictions due to the topographic mapping that underlies the GTM learning.

1.3 Author's Contributions

Publication I: OP-ELM and OP-KNN in Long-Term Prediction of Time Series using Projected Input Data

This journal paper proposes a modification to the variable selection problem, where a special projection matrix is used to project the data into a new space where the model is able to reach lower training error. The criterion to be optimised is the Delta test with the Genetic Algorithm acting as a search method. The complete methodology is applied on two time series competition data sets plus one bankruptcy prediction task. Both time series tasks are cast into the long-term prediction mode where the two models with fast training times are employed – Optimally Pruned Extreme Learning Machine and Optimally Pruned k-Nearest Neighbour. The author proposed the idea of the special projection matrix, performed the experiments involving variable selection, i.e., projection, and wrote the part related to that specific task, while the other authors contributed in their respective domains.

Publication II: Approximate k-NN Delta Test Minimization Method using Genetic Algorithms: Application to Time Series

In this journal paper, variable scaling and projection are further addressed where the nearest neighbour calculation involved in the Delta test is replaced with the approximate version to improve the computation time, while losing only small percentage of the accuracy compared to the exact version. The methodology is applied to many well known time series, but the approach is easily generalized to all regression problems. The idea

in this joint work was proposed by Fernando Mateo, who wrote the majority of the paper, while the author performed most of the experiments and wrote the part related to the complete setup of the algorithms and parameters.

Publication III: Climate Induced Changes in Benthic Macrofauna – A Non-linear Model Approach

This contribution is a joint work with the researchers working at the Leibniz Institute for Baltic Sea Research in Rostock, Germany. The publication is the result of interdisciplinary collaboration where the main goal is to discover which combination of the external factors influence the benthic species in the North Sea. This multifactor approach is an improvement over single factor (climate index) that is mainly used in the marine biology domain. The relationship is modelled with the Extreme Learning Machine neural network, and the author proposed a slight modification for the model to suit the small sample data for the task, performed all the experiments and wrote the part of the paper related to machine learning, while preprocessing steps are jointly discussed between the authors.

Publication IV: Multistart Strategy Using Delta Test for Variable Selection

This conference paper discusses how the optimisation landscape is formed when using the Delta test as a search criterion in data sets with large number of samples. That is, performing variable selection in large sample data produces a landscape where only few local minima exist, and as such the greedy forward-backward procedures are sufficient for the task if the algorithm is applied many times from random starting solutions. The proposed multistart strategy enables to restart from more promising solutions rather than random positions. The paper has a single author.

Publication V: Relevance Learning for Time Series Inspection

This publication deals with both variable scaling, or relevance learning, and time series inspection. The common approach in time series prediction is dividing the series with a sliding window, and the question remains how many and which lags contribute most to the series dynamics. The method proposed in this paper is an extension of relevance learning based on the Generative Topographic Mapping (GTM) adapted for the regression task which in turn returns an interpretable relevance profile. Another advantage of a GTM based approach is a prototype based learning which gives an easy and reliable way of making long-term predictions.

The proposed method is a joint work with all other authors, where the author performed part of the experiments and wrote the part related to data sets used in the experiments.

Publication VI: Extending Extreme Learning Machine with Combination Layer

This conference paper introduces a different approach to model structure selection for the Extreme Learning Machine neural network. One of the themes in the neural network literature are the small improvements to the basic single feed-forward model, either in the form of appropriate selection of hidden neurons or making the input weight distribution or the neuron output distribution more adaptable for the data. The proposed solution is to completely replace the model structure selection procedure (choosing the suitable number of neurons) with a weighted average over all possible neural structures from the pool. This model combining approach coupled with the Jackknife Model Averaging method and the leave-one-out residuals provides in some cases substantial improvement over a single model approach. The idea is discussed with Amaury Lendasse, while everything else is done by the author.

Publication VII: Variable Selection in a GPU Cluster using Delta Test

This publication deals with the implementation for variable selection where the Delta test is the criterion of choice in the heterogeneous cluster of computers. The concerns covered are the computation of the nearest neighbours and the distribution of solutions between the nodes in the cluster to achieve faster exploration of the solution space. An island model of the Genetic Algorithm with redistribution policy is used to effectively exploit all available computing power with an efficient mechanism to enable that both fast and slow machines are synchronised as much as possible. The author contributed in discussions about the publication while most of the work and writing was done by Alberto Guillén.

Publication VIII: Evolutive Approaches for Variable Selection Using a Non-parametric Noise Estimator

This book chapter provides a summary of different parallel paradigms in different hardware architectures. All paradigms use the Delta test for variable selection. Issues about the optimisation and efficiency are discussed when the search is performed on a cluster of computers (heterogeneous or homogeneous). The other aspect is that of global search over the solution space versus the small local improvements and how to bet-

ter incorporate both of these two aspects of search. Genetic Algorithm is used for the global search while the Tabu Search is employed for the local refinements. The author is responsible for implementing and writing the part related to Tabu Search refinement, while the part related to parallel implementation and Genetic Algorithm is done by Alberto Guillén.

1.4 Structure of the Dissertation

The remainder of the dissertation is divided into four chapters. The topic of variable selection is discussed in Chapter 2 along with proposed solutions using the Delta test. These solutions include different search algorithms, parallel implementations on clusters of homogeneous and heterogeneous architectures to speed up computation and further reductions with the approximate nearest neighbours search. Chapter 3 describes time series prediction problem with commonly used models and approaches to solve the problem. Novel models for time series prediction are also discussed in this chapter. These include two models based on the Extreme Learning Machine which can be applied to the standard regression data sets as well, and one model based on the Generative Topographic Mapping with a ranking layer for features. Applications to the real biological time series and connections with the climate factors are discussed in Chapter 4. Finally, conclusions and discussion about the future directions are presented in Chapter 5.

2. Variable Selection

This chapter discusses the variable selection problem in machine learning, starting from the basic principles to several solutions proposed in the publications. Two main aspects of the problem are explained: the *criterion* of evaluation and the *search algorithm* to explore the search (state) space. Both aspects are being developed independently where most of the search algorithms are coming from the domain of function optimisation, but can be readily applied to the problem of variable selection. From the many available criteria for variable selection, special attention is devoted to the *Delta test* [13, 14], a noise variance estimator that is useful before the modelling stage.

Several approaches to variable selection are discussed, from the *pure* selection method to the extension in the form of *scaling* and *projection*. The extended versions offer more flexibility, but suffer from the large search spaces taking considerably more time to find good solutions.

In the field of optimisation there are numerous algorithms for finding solutions to a function minimisation (maximisation) problems, and here we only discuss two among them. One is a meta-heuristic method Tabu Search [15, 16], developed for numerical optimisation, while the other is more widely known Genetic Algorithm [17,18] borrowing the idea of evolving a population of individuals or solutions. The latter one is particularly flexible for the parallel architectures which are used for a pure variable selection problem.

2.1 Description of the Variable Selection Problem

Machine learning methods and algorithms are applied to the data, which come in the form of samples and features in a matrix representation. If we denote with $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, N$, the data samples where

$$\mathbf{x}_i = (x_i^1, x_i^2, \ldots, x_i^d) \tag{2.1}$$

are the input variables for the data vector (sample) i, the standard way to model the relationship between the input vector $\mathbf{x}$ and the output y is to assume a functional form, i.e.,

$$y_i = f(x_i^1, x_i^2, \ldots, x_i^d) + \epsilon_i = f(\mathbf{x}_i) + \epsilon_i \tag{2.2}$$

where ϵ_i is the noise term. Both the input vector $\mathbf{x}_i$ and the target y_i are assumed to contain real values, i.e., $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. For further discussions we also use $\mathbf{X}$ to denote the complete input space or data space, that is, a matrix where rows are data samples and columns are variables. The data is said to have N samples and of dimensionality d, or with d variables. Similarly, $\mathbf{Y}$ indicates a column vector of outputs and its dimension is $N \times 1$. This can be written as follows:

$$\mathbf{X} = \begin{bmatrix} x_1^1 & x_1^2 & \ldots & x_1^d \\ x_2^1 & x_2^2 & \ldots & x_2^d \\ \vdots & \vdots & \ddots & \vdots \\ x_N^1 & x_N^2 & \ldots & x_N^d \end{bmatrix}, \qquad \mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

and the Eq. (2.2) can be rewritten in matrix form as follows:

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_N) \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_N \end{bmatrix}$$

or more simply as $\mathbf{Y} = f(\mathbf{X}) + \boldsymbol{\epsilon}$. This setup is regarded as a *supervised learning* approach since the labels, or in this case real values $\mathbf{Y}$, of the target variable are known in advance. When such information is lacking, the problem is categorised as an *unsupervised learning* problem. For the rest of the chapter we are only interested in the supervised version as given by Eq. (2.2). Since $\mathbf{Y}$ is a vector of values, this is a univariate case of regression and we are only concerned with this type of the problem ignoring the multivariate case. In the multivariate case, the output for each data vector $\mathbf{x}_i$ is another (output) vector $\mathbf{y}_i$ instead of a scalar y_i.

Many machine learning methods assume that all d variables are relevant to be able to predict y, but this is not always the case. For example, both k-Nearest Neighbour and Support Vector Machine [19] models have

inherent assumption that all features are important or relevant to establish a relationship between the inputs and the target variable. For Support Vector Machines this depends on the kernel function, but the most widely used symmetric Gaussian kernel takes into consideration all input features. First, there can be situations where certain variables are redundant, that is, information contained in those variables is already present in other variables. A simple example is having two highly correlated variables in the data set, or the extreme case of identical features where they are equal across all samples. These special cases exemplify that certain precautions must be considered and that not all variables are important just because they are included in the data. The second reason for variable selection lies in the limited amount of data samples which has significant impact on the learning algorithms. Many machine learning methods contain what are called *hyper-parameters*, parameters governing other primary parameters, for example, the noise variance ϵ. In order to have stable and reliable estimates of these hyper-parameters, a large number of data vectors is needed. On the other hand, limited amount has negative effect on the learning process in the form of *overfitting*. Overfitting, or overtraining, is a situation where the model is able to *fit* the data to a high degree of accuracy if not perfectly, that is, the prediction $\hat{y}_i$ is (almost) equal to the target y_i. The downside of this case is poor generalisation capability of the model where subsequent predictions for unseen samples significantly deviate from their true values. Overfitting usually occurs when the model has too many adjustable parameters which are fitted not only to the data, but also to the random fluctuations (noise) [1,20]. The opposite case is *underfitting* – the model is too rigid with a few parameters and is not able to model any kind of nonlinear behaviour. One way of combating this problem is to reduce the number of dimensions d. This is closely related to the *curse of dimensionality* [10,21], where the number of features d greatly outnumbers available samples N, and where building a model becomes a difficult challenge. The third issue is related to the execution time, as with less features certain models can be trained faster giving practitioners a quicker insight into the input–output connections. The last issue is related to the complexity and interpretability, where a simple model is much easier to understand (a model containing only a couple of input variables) against a more complex model with many variables included.

Another way of achieving a lower dimensional representations of data is

via *data compression*. In this approach, data samples $\mathbf{x}_i$ are transformed into the vectors in the lower dimensional space $\bar{\mathbf{x}}_i \in \mathbb{R}^p$ ($p < d$) either with a linear or nonlinear transformation. One of the most popular techniques in machine learning for reducing the dimensionality of the data and eliminating certain levels of noise is the Principal Component Analysis (PCA) [22]. Each new vector $\bar{\mathbf{x}}_i$ is a linear combination of the original variables x_i^j, $j = 1, \ldots, d$. Main advantage of the PCA is that it requires no additional parameters, except the decision to how many dimension p one wishes to project to. Another important aspect of the PCA is that the solution is unique, that is, it does not have any random elements or initialisation issues as some other methods. The main drawback is its linearity which is not suitable for most real-world problems and the interpretation of the newly created feature space since the original variables are lost with this transformation.

More formally the goal of variable selection can be described as follows. Denoting with X^j the j-th variable in the data set, and with X^S a subspace of $\mathbf{X}$ containing only $S = \{j_1, j_2, \ldots, j_k\}$ variables, the goal is to find a subset $S \subseteq \{1, 2, \ldots, d\}$ such that the model built using variables in S provides us with the best possible generalisation ability out of all possible subsets with d features. Indices $j_i, i = 1, \ldots, k$, denote the variables X^{j_i} in the data. The notion of "the best" subset is usually measured with a specific criterion, called relevance or search criterion, while the traversal over the possible subsets is done with a *search algorithm*. These two distinct parts constitute the variable selection procedure and for the most part are independent of each other. This enables different combinations of criteria and solutions offered in the field of optimisation algorithms. The problem of variable selection has been studied both in statistics [23, 24] and in machine learning community [25, 26].

In machine learning, the central part of any inference procedure is a model, and this has further implications on the variable selection. On a general level, variable selection strategies can be categorised in two groups: filter and wrapper approaches [25].

Filter Strategy

Filter approach [27, 28] aims to find an optimal subset of features before the learning phase of a model. The choice of a search criterion is even more pronounced in this strategy since once the final choice is made it cannot be revised. Many criteria can be applied here, from the basic cor-

relation coefficient [29], mutual information [30–32] and data-driven approaches [33]. Filter methods provide a fast way of finding promising feature subsets, but their performance can be poor if the learning technique has a different goal compared to the search criterion in the first phase. This is often the case, as learning methods have some form of a training error as a performance measure (usually least-squares or mean-square error), while certain criteria have other measures. For example, mutual information measures how much information is contained in one variable to explain the other using the notion of entropy of a random variable.

Wrapper Strategy

Different from the filter approach, wrapper strategy [25, 34] uses some form of the learning algorithm to assess how efficient is a specific subset of variables. For each subset under evaluation, a model is build and is compared to the currently best solution found up to that point. This is the potential pitfall of the wrapper approach as it requires training a model multiple times and can lead to substantial computational cost. Certain models provide an extremely fast training times and reliable estimation enabling wrapper approach to be a viable strategy even with moderately large data sets. The advantage of the wrapper method is reliable estimation of the underlying connection between the inputs and the output, since the same learning method is used in both stages: finding a good subset and the final training phase. Usually some form of a validation procedure is used to guarantee good generalisation properties of the model.

2.2 Search Algorithms

The purpose of a search algorithm is to traverse the solution space in order to find the optimal solution. Solution space is the set of all possible solutions to the problem, and is also know as state space, objective function space, or optimisation landscape. Many different approaches are possible and over the years many methods have been proposed, such as Genetic Algorithms [25, 35, 36], Monte Carlo Markov Chain [37], greedy local strategies [26, 38], Greedy Randomised Adaptive Search Procedures (GRASP) [39, 40], Differential Evolution [41], Evolutionary Algorithms (EA) [42] which include many nature inspired techniques such as Ant Colony Optimisation [43].

Formerly, many search algorithms could not be applied to the practical

problems having many variables due to their long computational times. This trend has shifted a bit since the advent of the parallel architectures and the Graphics Processing Unit (GPU) computing. Unfortunately, the size of data sets has grown substantially in the recent years in the form of so called "big-data", and the problem of traversing the search space comes down to clever utilisation of visited regions of the space.

Several widely used approaches for the variable selection are discussed in the following sections – exhaustive search approach based on enumeration, greedy strategies in the form of Forward and Backward Searches, and the advanced solutions based on the Tabu Search and the Genetic Algorithms.

2.2.1 Exhaustive Search

The simplest approach is to examine all the possible subsets that can be formed on the set $\{X^1, X^2, \ldots, X^d\}$. This requires evaluating $2^d - 1$ solutions in total and can be computationally too demanding even for the moderately large data sets with $d > 30$. Even though this yields the optimal solution among all possible candidates, it is prohibitive to use this approach for the cases where dimensionality exceeds several hundreds of features. Image, text, video databases, chemometrics data that contains information over a wide spectrum of frequencies, gene expression data in bioinformatics all have several thousand variables which prevents using any kind of an algorithm for finding the optimal solution since the state space is enormous. Therefore, a more practical strategy is required in order to find some feasible solution, i.e., a local optimum, which is not necessarily globally optimal, but nevertheless is the best among its *neighbouring* candidates. In certain scenarios, these locally best solutions are sufficiently good and acceptable in practice.

2.2.2 Greedy Search Algorithms

Forward-Stagewise Linear Regression

The idea behind the greedy algorithms is again a simple one, and it involves searching only a small subset of all the possible solutions. This approach has been used in statistics for a long time in the form of Forward-Stagewise linear regression [44] where the inputs are added according to the correlation coefficient of the candidate (remaining) features with the current residual vector. The feature with the highest correlation is the one

added to the set of the *best* features (previously included in the set). This has been later extended into the Least Absolute Shrinkage and Selection Operator (LASSO) [45] and the Least Angle Regression (LARS) [46]. The main goal is to find an acceptable solution in a small amount of time, and these procedures sacrifice optimality for the execution time.

Although intuitively appealing and fast, the downside is that only a small portion of the state space is explored in some local region or neighbourhood. For this reason these methods are sometimes called *local* search procedures. This is the major problem of greedy strategies, since they stop after finding the local optimum. In order to explore a wider area of the state space, several initial solutions are generated, then from each solution the variable selection procedure is applied until a local optimum is found, and finally the best solution among these local optima is chosen to be the final solution to the variable selection problem. In the remainder of this section, we describe several commonly used greedy strategies for variable selection.

Forward Search

Forward Search follows the same principle as the Forward-Stagewise regression. We start with an empty set S and keep adding one variable at a time into S based on the search criterion, that is, we add the variable which brings the most information (the most predictive power) to the current solution S. This process is repeated until the relevance criterion no longer improves over the previous solution. Following this procedure, the total number of calls to the search criterion is at most $d(d-1)/2$ and this number is attained when the best solution has all d variables. Alternatively, we can keep adding variables until all of them are included in S which provides a ranking according to the selected relevance criterion. Algorithm 1 gives the outline of the Forward Search strategy that includes all variables, but returns the best solution found among the generated subsets. The strategy assumes that minimisation is taking place, while the problem of maximisation can be easily modified to fit into the same strategy.

In Algorithm 1, $\mathrm{Criterion}(S, \mathbf{Y})$ computes the relevance criterion between the set of input variables S and the output vector $\mathbf{Y}$. Even in the case when all the variables are included in the set S, the procedure gives vastly reduced number of evaluations compared to the exhaustive search.

Algorithm 1 Outline of the Forward Search algorithm

1: set $S_0 = \{\}$ and $U = \{X^1, \ldots, X^d\}$ {initialisation}
2: **for** $k = 1$ **to** d **do**
3: $X^j = \arg\min_{X^i}\{\text{Criterion}(S_{k-1} \cup \{X^i\}, \mathbf{Y}) \mid X^i \in U\}$
4: $S_k = S_{k-1} \cup \{X^j\}$ {inclusion step}
5: $U = U \setminus \{X^j\}$
6: **end for**
7: **return** $S = \arg\min_{S_k} \text{Criterion}(S_k, \mathbf{Y})$

Backward Search

Backward Search proceeds in the opposite directions compared to the Forward Search starting from the full set of variables $\{X^1, X^2, \ldots, X^d\}$ and then removing one variable at a time. Both procedures share the same properties: they return the local optimum and the total number of evaluations is at most $d(d-1)/2$. The computational times for these two procedures can be quite different since the Forward Search can benefit greatly from evaluations in much smaller subspaces with only couple of features compared to the full data that the Backward Search starts with. For example, any nearest neighbour approach needs to compute the differences across all the features which consumes much more resources compared to the calculations in subspaces with couple only of features.

Forward-Backward Search

Comparing both the Forward and the Backward Search procedures they also share the following property – in the inclusion (exclusion) step, the variable added (removed) only contributes toward the search criterion with all other variables currently in the set S. If the variable X^j is included in set S, then both procedures only test the combination $\{X^j\} \cup S$ while combinations of X^j with the subsets of S are ignored. Forward-Backward Search (FBS) tries to mediate this issue and explore a more wider space compared to both Forward and Backward Searches. The idea is to try both *selecting* variables that are not in S, and *discarding* those that are already in S. In all the iterations in this approach, a total of $d-1$ solutions are checked before the best subset is chosen. In principle, the algorithm can start from any solution, not just an empty or the full set of variables. This implies that the number of evaluations is unknown due to the unknown starting position in the solution space. The complete algorithm is given as Algorithm 2.

Algorithm 2 Outline of the Forward-Backward Search algorithm

1: set $S_0 =$ random non-empty subset
2: $U = \{X^1, \ldots, X^d\} \setminus S_0$
3: **for** $k = 1$ **to** d **do**
4: $\quad X^j = \arg\min_{X^i} \big(\{\text{Criterion}(S_{k-1} \cup \{X^i\}, \mathbf{Y}) \mid X^i \in U\} \cup$
$\qquad\qquad \{\text{Criterion}(S_{k-1} \setminus \{X^k\}, \mathbf{Y}) \mid X^k \in S_{k-1}\}\big)$
5: $\quad S_k = \text{update}(S_{k-1}, \{X^j\})$ {add or remove}
6: $\quad U = \text{update}(U, \{X^j\})$ {remove or add}
7: **end for**
8: **return** $S = \arg\min_{S_k} \text{Criterion}(S_k, \mathbf{Y})$

The update operator in the Algorithm 2 modifies a variable set according to either the removal or inclusion of a chosen variable X^j.

The advantage of the Forward-Backward approach is the ability to start from any position of the solution space compared to a fixed initial position used in both Forward and Backward Searches. If prior knowledge about the important variables is available, this can be used to form a starting position for the FBS. This enables Forward-Backward Search to find many local optima provided that several initial positions are generated, and this strategy can be quite powerful when the Delta test is used as a search criterion, see Publication IV.

2.2.3 Encoding the State Space

In variable selection, the feature can only have two possible states: present in or absent from a subset. This naturally leads to the binary encoding of a solution using the indicator variables $I^j \in [0, 1]$. An indicator vector $\mathbf{I} = [I^1, I^2, \ldots, I^d]$ represents one solution in the *encoded form* corresponding to the actual solution S, i.e.,

$$I^j = \begin{cases} 1 & \text{if } X^j \in S \\ 0 & \text{otherwise} \end{cases} . \tag{2.3}$$

The encoded version is used when implementing aforementioned algorithms, but it also plays an important role when defining the *neighbourhood* of a solution on which certain strategies rely. Neighbourhood structure can be defined in many ways, but the representation already described suggests the following strategy. For a given indicator vector $\mathbf{I}$, its neighbouring solutions $Ne(\mathbf{I})$ (subsets that are going to be examined next) are the ones where each candidate has one feature inverted compared to

I, i.e.,

$$Ne(\mathbf{I}) = \{\mathbf{I}_c^j \,|\, \mathbf{I}_c^j = [I^1, \ldots, I^{j-1}, |1 - I^j|, I^{j+1}, \ldots, I^d],\, j = 1, \ldots, d\}. \quad (2.4)$$

This structure and connections between the solution **I** and its neighbours are known as the d-dimensional hypercube in graph theory. Neighbourhood structure plays a crucial role in the following section which describes the Tabu Search.

2.2.4 Tabu Search

Tabu Search (TS) belongs to the family of *meta-heuristic* methods whose main purpose is to employ other local search algorithms in order to avoid local optima. Developed in the late 1980's in the field of combinatorial optimization by Glover [15, 16, 47], it has been extended to different domains ranging from scheduling problems [48], routing [49,50] and general optimisation problems [51, 52].

In the field of optimisation, the main goal is to find a global optimum to a specific cost or objective function, and here we assume that the problem is defined with a function F and the goal is to minimise F, i.e.,

$$\min\ F(S)$$
$$S^* = \arg\min_S F(S)$$

with S^* being the global minimum to the problem F. While global optimality cannot be guaranteed, many methods try to explore the search space as much as possible in order to find more promising regions and accordingly refine the search to pinpoint the actual local optimum. For variable selection, the objective function is just another term for a relevance criterion, and these names are used interchangeably in this section. In this thesis, we are using the same notation S for both the subset of variables and a solution to the optimisation problem. The reason for this is that a subset of variables is one solution to the variable selection problem, while the problem itself falls into a broad class of optimisation problems.

The term meta-heuristic refers to the underlying idea of TS – it uses other technique, denoted T, for the search through the solution space. During the search, TS uses internal memory structures to modify the way T visits solutions. One of the main purposes of the memory is to prevent

the reversal of the recently applied moves, but also to reinforce the exploration of promising areas of space. These two types of memory can be broadly categorised as the short-term and long-term memory respectively. The idea behind TS is to use the technique T until it reaches a local optimum, but contrary to the greedy procedures, it does not stop once the optimum is reached and proceeds to visits solutions with the worse objective values. The memory structures keep track of the local optima and the technique T is forbidden to revisit these in the upcoming iterations. This concept of moving towards solutions which do not improve the objective function has roots in Simulated Annealing [53] where the acceptance rate is based on a stochastic mechanism tightly connected with a cooling scheme. This approach is also present in Markov Chain Monte Carlo (see [37] or [1, Chapter 11]) techniques with a property that a move is made towards new a solution if the acceptance step is fulfilled.

Next important issue is the definition of the neighbourhood of a solution. The neighbourhood is defined as the set of solutions that are *reachable* from the solution $\mathbf{I}$. Reachability is defined through moves or local transformations applied to $\mathbf{I}$ in order to produce solutions in $Ne(\mathbf{I})$ as explained in Section 2.2.3.

The short-term memory is responsible for preventing the cycling effects and it keeps track of the recently applied moves. Sometimes it is referred to as the *tabu list*. Once the (sub)optimal solution has been found, the tabu list forbids the search to revisit this solution by restricting the use of a move with the reversing effect. Moves stored in the tabu list are called tabu, and thus forbidden to use for a fixed number of iterations. Storing only moves does not guarantee the prevention of cyclic effects, as not all information is kept when the optimum has been found. To stop revisiting, one can store *complete* solutions in the memory. However, this approach becomes impractical as the complexity of the problem increases, and the substantial amount of execution time is spent on comparing new solutions to the ones stored in memory. This is the reason for keeping smaller pieces of information, such as moves, segments or other attributes of solutions, for example, the binary encoding in a specific range of variables.

One important parameter of the TS is the *tenure* [54, Chapter 2.4], which is defined as the number of iterations a single move is considered tabu. In some implementations this corresponds to the length of the tabu list, usually coded as a cyclic list. The tenure value is fixed throughout the whole search for most of the problems, but other approaches are possible:

varying tenure value or randomly choosing the value for each move.

During the search, the tabu list can prevent the moves to solutions which have not been encountered before (assuming no storage of complete solutions). This leads to a situation when the TS discards a move to a solution with the better objective value than the currently best one. To enable such moves, another level is added to the TS, which allows the search to override the tabu list and *aspire* to the new solution. This is known as the aspiration criterion. The simplest aspiration criterion is to allow the move in the case just described, when the best solution so far has been found. Other criteria can be defined to revoke the tabu status, but they are seldom used. In the implementation of the TS in the publications, no aspiration criteria are used as they involve computing the actual objective value of a solution. By dropping these criteria, more computational resources are devoted to new solutions allowing more exploration of the search space in the same amount of time.

Like all searching algorithms, the TS does not guarantee global optimality and thus it is impossible to define for how long it should explore the solution space. Therefore, certain stopping condition must be defined to prevent the methods from running indefinitely. Several choices that are used throughout the literature include: amount of time spent for the search, number of iterations, total number of calls to the objective function and other heuristic possibilities, out of which the first two conditions are the most commonly employed ones.

There are other parts of the TS which make it a powerful method, such as probabilistic TS (using stochastic acceptance rate to further help jump out of local optima), candidate list generation, intensification and diversification strategies, and auxiliary objectives to name a few. These are not considered here, but for a detailed explanation on the topics see [54].

2.2.5 Genetic Algorithms

Genetic Algorithm (GA) is one of the algorithms of the larger family of optimization techniques known as the Evolutionary Algorithms (EAs) [55]. Most of these algorithms are based on the concepts inspired by the nature, such as GA being based on the evolutionary mechanisms, while the Ant Colony Optimisation algorithm has foundation in the behaviour of the ant colonies and their mutual underpinnings. Certain algorithms are *population* based, that is, a pool of solutions exists at all stages of the execution. This pool is sometimes randomly initialised while other *operators*

underlying the technique(s) also have a degree of randomness involved which suggest that most of the EA algorithms are stochastic search procedures. Among the common operators among EA algorithms are: encoding of the solutions as *chromosomes*, initialising the first pool of solutions – the initial population, selection operators, and reproduction operators. We briefly describe the basics of the Genetic Algorithm, alongside the concepts developed for the parallel implementations and the multi-objective optimisation problems. More formal treatment and theoretical background can be found in [17, 18, 56].

Contrary to the classic optimisation techniques based on the first and the second order derivatives where the idea is to make *deterministic* moves to explore the solution space, evolutionary based algorithms (including the GA) use the function values, also called *fitness* values, and probabilistic rules to examine the optimisation landscape. Comparing the GA with both the TS and the FBS, both the TS and the FBS do not need information on the function shape, but they are neighbourhood based – FBS explicitly, while the TS relies on some local search procedure that also needs the neighbourhood structure. On the other hand, the GA uses a population of solutions enabling the search of several areas of the objective space and can potentially find promising areas much quicker than TS and FBS.

Genetic Algorithm Basics

Algorithm 3 shows the general idea of the GA while the description of each of the operators is explained afterwords.

Algorithm 3 Outline of Simple Genetic Algorithm

1: select *selection* operators ν_1 and ν_2
2: select *reproduction* operators ρ
3: Π = create initial population
4: **while** not *stopping condition* **do**
5: $e = \text{fitness}(\Pi)$ {evaluate population}
6: $\Pi_1 = \nu_1(e)$ {select parents}
7: $\Pi_2 = \rho(\Pi_1)$ {reproduction – generate offspring}
8: $\Pi = \nu_2(\Pi, \Pi_2)$ {select new generation}
9: **end while**
10: **return** solution π with the best objective value from Π

Representing solutions. Each solution to the optimisation problem represents one individual in the population. An individual in turn is given as a *chromosome* which encodes all the characteristics of a solution. In the domain of variable selection, these characteristics are simply indicators of presence/absence of each variable, and they are called *genes*. A chromosome is simply an indicator vector $\mathbf{I}$ of length d, and each gene correspond to one variable in the data set. For the rest of the chapter, we interchange the names of a solution, chromosome and individual since indicators (genes) in a chromosome fully explain the individual, which is the solution to the problem.

Initial population. As outlined in Algorithm 3, the GA is based on modifying its population Π of solutions. In most optimisation problems, this population is initialised randomly with the idea of covering the search space as evenly as possible. For this reason, the uniform initialisation is suitable in most cases, although specific problems require more appropriate creation in order to accommodate any prior knowledge. Covering most of the solution space enables the GA to find more promising regions and through reproduction create the combinations which should contain even better solutions (this is the idea behind the schema based GA [57]). Population size is the parameter that needs to be decided before the optimisation process. There is a trade-off between the size of the population and the *convergence speed* of the algorithm. Larger populations enable better spread of the solutions across the state space, and less iterations are needed for the GA to converge. The other possibility is to reduce the population size, speeding the process and providing a good solution a within reasonable time, but generating new promising regions is severely hindered. Population size impacts the execution time, and the more individuals there are, the more time is required to generate the new population (the next generation). Convergence of the GA is considered as the situation where the whole population is dominated by just a few individuals.

Evaluation of the individuals. Evaluation is a simple step, which involves computing the objective function value for each individual in the population. These values are used in the first selection phase which is followed by the reproduction step. Since different optimisation problems have distinct values of the objective function, the usual practice is to scale these objective values to a more suitable range. This is the purpose of a *fitness function* which returns fitness values for all individuals. Most of the time the fitness values are confined to a $[0, 1]$ range and treated as probabili-

ties of selecting an individual for the reproduction phase. In this setting, higher fitness value indicates an attractive solution which should be kept for the reproduction, that is, selecting the same solution with a higher probability.

Selection. The purpose of the selection operator is to emphasize more promising solutions over the bad ones (in terms of the fitness value). Selection takes place at two points during the execution of the algorithm:

- selection of individuals for the reproduction (operator ν_1 in Algorithm 3)
- selection of new individuals for the next generation (operator ν_2)

New individuals, also called the offspring, are created by applying the *crossover* mechanism and/or *mutation*. Following that terminology, the solutions from which the offspring are created are named parents. The main point of the crossover is to recombine the genes of the individuals for finding either more fit individuals or exploring new regions of the solution space. On the other hand, mutation changes the genes without considering the fitness value of the individual. This is to ensure that small variations from the best solutions are always present in the pool to avoid too fast convergence. Convergence in the GA domain is tied to the diversity of the population, and when the population is dominated by relatively small number of similar individuals, the algorithm is considered to have converged. However, the mutation can also have negative effects if the scheme and the frequency of mutations is set up inappropriately. The search in this case can lead to a random walk behaviour.

The second selection operator functions after the creation of the offspring and the purpose is to choose the individuals based on both the parent set and the offspring set. Selection can choose from solutions present only in the offspring set or from both sets. The scheme when only the offspring set is taken into consideration is called the *replacement* strategy.

Selection operators are often characterised by their *selective pressure*, which is defined as the speed at which the best solution will occupy the entire population by repeated application of the selection operator alone. If this pressure is too high, the algorithm produces similar, if not exactly the same solutions, and loses diversity of the solution pool which hinders the exploratory abilities. With low selective pressures, the diversity is preserved but at the expense of the slower convergence.

Selection operator called *elitism* is one of the widely used selection oper-

ators. The main idea of elitism is to simply copy the best individuals from the parental set into the next generation. With this approach, during the execution of the GA, the fitness value never deteriorates and prevents the drift caused by the random mutations.

Reproduction. In the reproduction step, the new individuals (offspring) are generated based on the current pool of solutions (parents). Crossover operator is responsible for creating new solutions by recombining the genes of the parents. The premise is that both parents contain "good" genes for specific regions of the search space, and that their combination produces a solution having all the good genes. Second stage in the reproduction is the mutation step (although not mandatory). With mutation, small alterations are performed on the genes to introduce more diversity and keep the solution pool from stagnating.

Based on the encoding of the solutions, crossover operators can be categorised into two classes: binary and real-valued. Binary encoding is mentioned in Section 2.2.3, while real-valued approach is used when the *variable scaling* approach is adopted, and this is explained in Section 2.3.1. Important issue in the reproduction stage is the *stochastic* nature of the crossover step, that is, recombination is performed with a certain probability. The probability is set to a high enough value to ensure that new individuals are created, but this method also guards against quick drifts from the best solutions caused by a weak version of the elitism selection operator.

Stopping conditions. All of the previously mentioned operators are applied in each generation until a predefined stopping condition is met. The simplest approach is to put a limit on the number of generations created by the GA. Other conditions involve the time constraint, testing the convergence of the algorithm, diversity measurements among the individuals [58], no change in the objective function value for a number of consecutive generations, and many others.

Parallel Implementations

Genetic Algorithm uses the operators on a population of independent solutions, and this observation makes it easy to adapt and distribute the workload on parallel architectures. The distribution of the population is the first step in implementing the GA for a particular problem. Population can still remain as a single set of solutions, or it can be divided into several smaller groups which is also called the GA with multiple populations

(depending on the point of view). In the multiple population approach, another consideration must be taken into account – communication policy between populations. Populations can be completely separate, and this approach is just a GA with different initial populations with the difference that the algorithm is executed multiple times in parallel. If there is communication between the populations, the adopted policy must specify the pattern of communication: the number of individuals to exchange and the rate of exchange, which incurs additional burden on available resources in the network. The exchange should not overtake the computational resources away from the core operators of the GA and must be kept at a minimum.

Parallel GA allows dividing a larger problem into smaller ones, where each small problem is tackled on a unique processor. Among the many possibilities for this division, the most common classification of parallel GA is the following: single-population master-slave GA [59], multiple-population GA, fine-grained GA, and hierarchical GA. We briefly explain the first two categories that are relevant for Publication VII and Publication VIII.

Master-slave single GA. In the master-slave topology, one *master* processor or node is responsible for the sequential part of the GA – selection and reproduction, while all the other nodes or *slaves* are devoted for evaluating the fitness values of the individuals. Communication consists of sending individuals from the master node to the slaves (first direction), and after evaluation, the slaves send the computed fitness values back to the master node to be used in the core part of the GA. This approach is still a simple GA with only one population where the workload is spread among the available resources. Two extra steps are added to Algorithm 3: one before (master to slave) and one after (slave to master) Line 5 that involves evaluation of the individuals.

Multi-population GA. Contrary to the master-slave approach which still retains a single population, in the multi-population GA the pool is divided into groups, also called subpopulations. These subpopulations exchange information by migrating some of the individuals between themselves. This policy is controlled by several parameters: the rate of migration – at which points of the algorithm the migration occurs, the number of migrating individuals, the pattern of migration – the source and destination of one pair of nodes in the policy with multiple choices available, and the selection of individuals for migration. The multiple-population parallel GA

is also know as the *island model*, since small subpopulations resemble separate islands and possible interconnections between them.

Synchronisation schemes. Another important aspect in the parallel implementations is the synchronization scheme which can be classified into two types: synchronous and asynchronous. Both of these types are applicable to all parallel categories of the GA. In the synchronous approach, all the processors have the same population at their disposal, while the communication exists in order to synchronise the processes. This leads to the scenario where faster machines have to wait for the slower ones to finish their respective jobs. Once all the jobs are executed, the algorithm continues onto the next generation. The asynchronous case is designed to prevent such situations, that is, all the machines execute the code without any delay resulting in much less idle time.

Genetic Algorithm for Multi-Objective Optimization

Multi-objective (MO) [60–62] optimisation is the case when at least two conflicting or competing objective criteria are present in the problem and all of them need to be optimised. In the case of two (minimisation) objectives, this is expressed as

$$\min_{S} \mathbf{F}(S) = \{F_1(S), F_2(S)\} \tag{2.5}$$

with the possibility of both equality and inequality constraints.

Instead of finding the global optimum as in the single objective function problems, the goal in multi-objective optimisation is to find the global *pareto-optimal set*. This set contains the solutions which are *non-dominating* with respect to each other in the objective function space, that is, where one solutions S_{i_1} has a better value for a particular objective function, say F_1, than another solution S_{i_2}, but solution S_{i_2} has a better contribution in a different objective F_2. In this case, one solution cannot be chosen over the other just based on the values F_1 and F_2, and the goal is to return both S_{i_1} and S_{i_2} (and possibly many other solutions) which form one of the pareto sets. The pareto-optimal *front* is a set of values in the *objective space* which are non-dominating. If the solutions S_i are members of pareto-optimal set, then their corresponding objective function values $\mathbf{F}(S_i)$ lie on a pareto-optimal front.

Once the pareto set is returned, one can choose a solution based on certain preferences. A simple example is that of quality to price ratio, where one objective is to reduce the cost (minimise F_1) and the other to

have higher quality (maximise F_2). Figure 2.1 shows this example with two pareto fronts (one local and one global), and three solutions (A, B, C) showcasing the concept of (non)dominance. Two solutions A and B lie on a global pareto-optimal front (line l_1) with A having better quality than B (maximise F_2), but suffering in price (minimise F_1). Third solution C belongs to a local pareto front, and is inferior to both A and B in both objective functions.

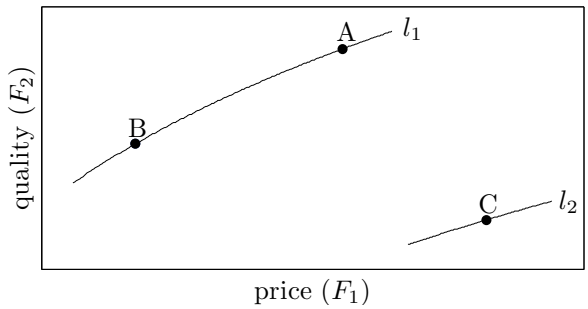

Figure 2.1. Global pareto-optimal front (line l_1) and local pareto front (line l_2). For further details refer to the preceding text.

Finding the global pareto-optimal front cannot be guaranteed, but the algorithms designed to tackle this problem must have two properties: generating solutions along the currently found pareto fronts and looking for the new pareto fronts.

For a GA approach which involves the reproduction phase, there is a possibility to construct the solutions which are dominated by other solutions in the population. They are discarded since there is at least one individual in the pool that dominates them, which leads to a problem how to efficiently generate non-dominated ones. The most widely used version for the GA based procedures is the Elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II) proposed in [63]. NSGA-II algorithm has two basic steps: sorting solutions based on the dominance factor and the elitism selection. The first stage is to ensure that the best individuals in terms of all objective functions are kept for the recombination phase to generate new solutions from the same pareto fronts. The second stage uses elitism in order to keep the pareto fronts intact and passed to the next generation. In the algorithm, the population size does not change, and the current number of solutions in several pareto fronts might not fully populate the next generation. For this reason, the least prominent front (the last one to be included) needs to be split since the interest is only in the set of *diverse* solutions to supplement the remaining positions

for the next generation. This is accomplished by using the *crowding distance*, a metric measuring the distance between the individuals in the objective function space, instead of the solution space. Most of the genetic algorithms for the MO have a hyper-parameter controlling the population diversity which needs to be chosen beforehand. NSGA-II does not have such a hyper-parameter making it an attractive method for the MO. Moreover, the diversity measurement is mainly focused in the solution space. The sorting in the NSGA-II is done by a careful book-keeping to speed up the execution time, but *both* the current population and the offspring are taken into account effectively doubling population size.

2.3 Variable Scaling and Projection

2.3.1 Scaling

Variable scaling tries to go beyond the pure selection problem. The indicator variables I^j provide the information about whether the feature is important or not, and all of the selected variables are given an *equal weight*. In the scaling approach, the goal is to provide a *ranking* with weights w with the usual interpretation that $w^{j_1} > w^{j_2}$ signifies that a variable X^{j_1} is more important than a variable X^{j_2}. This can be also seen in the case of simple linear regression $y_i = \sum_{k=1}^{d} w^k x_i^k$ where the coefficients indicate how much each variable contributes to the output y_i. However, this means that the coefficients are unbounded and the usual method is to constrain them to the $[0, 1]$ range where extreme values have the same interpretation as in the selection problem. In this setting, the scaling method is a more general problem than the pure selection.

Although scaling offers greater flexibility than the selection, the main drawback is the increased solution space to explore. Since the weights take values from $[0, 1]$ range, there are theoretically infinite number of solutions, but due to the numerical representation of the real values on computer hardware that number is finite albeit still massive. This fact alone prevents neighbourhood based algorithms discussed in Sections 2.2.2-2.2.4 to be used for this problem. One solution is to split the $[0, 1]$ interval into equally sized segments or subintervals, that is, to form a discrete neighbourhood structure. The newly created set of possible weights contains values that are equidistant from each other. For example, the division

can be done to form a set $H = \{0, 0.5, 1\}$. Usually, there is a parameter h that controls the number of subintervals. For the given example of $H = \{0, 0.5, 1\}$ we have $h = 2$ and $H = \{0/h, 1/h, 2/h\}$. Given a parameter h with a positive integer value, the set of weights is formed by forming $H = \{i/h \mid i = 0, 1, \ldots, h\}$. The set H will be referred to as a *set of weights* or a *set of scaling weights*.

With effectively discrete set of weights H, the neighbourhood structure can be defined in different ways according to the goal of the optimisation. For example, the neighbourhood can be defined as a set of solutions where the weight for a single variable X^j is modified to the closest value of w^j, i.e.,

$$Ne(\mathbf{w}) = \{\mathbf{w}_c \mid |w^j - w_c^j| = 1/h \wedge w_c^j \in H,\, j = 1, \ldots, d\}. \tag{2.6}$$

Another solution is to allow all possible changes for a single feature as

$$Ne(\mathbf{w}) = \{\mathbf{w}_c \mid w_c^j \in H \setminus \{w^j\},\, j = 1, \ldots, d\}. \tag{2.7}$$

The difference between these two definitions is the trade-off between exploration of the space and the execution time, as the size of the neighbourhood in the first definition is at most $2d$ while in the seconds definition it is hd. Those numbers indicate how many solutions are examined at each iteration during the descent/ascent phase. In the former case, the moves are made quicker but the examination of the state space is slower, while in the latter case each move takes more time while the exploration is much faster.

In the case of scaling the data set, each dimension is modified according to its weight factor

$$(x_i^j)^{\mathrm{S}} = w^j x_i^j, \quad i = 1, \ldots, N,\ j = 1, \ldots, d\,. \tag{2.8}$$

This way, a new data set is formed, which has at most d variables. The dimensionality depends on the scaling weights, and when all $w_i \neq 0$ the new data set $\mathbf{X}^{\mathrm{S}}$ has the same dimensionality as the original data set. On the other hand, it is possible to transform the data set using a linear projection, with the explicit specification of the dimensionality of the newly created data set.

2.3.2 Projection

Projection approaches are a commonplace in machine learning. The idea is to map (project) the data into a different transformed space which has desirable properties. One of the properties that had a significant impact on the learning algorithms is the linear separability for the classification task – finding an embedding in which two classes can be separated by a hyperplane. Different methods try to perform this specific task in a variety of ways, ranging from the Support Vector Machines and the closely related kernel methods, neural networks with transformations represented in the hidden layers and interconnections [4, 64] and the linear models with basis functions [1] to name a few. Many of these methods are based on the nonlinear projections where the goal is to have low training (misclassification) errors on the available data. As mentioned in Section 2.1 many of the algorithms take all the inputs as equally important which can lead to the overfitting issues.

While nonlinear approaches can provide flexibility and accuracy, linear projection based methods offer speed for the reduced representational capabilities. The most well know linear projection based method is the Principal Component Analysis (PCA) (see [22] or [1, Chapter 12]) where the goal is to find a linear subspace where the data retains maximum variance. A different perspective which provides the exact same solution is the minimisation of the mean-squared reconstruction error between the original and the latent spaces.

On the other hand, it is possible to explicitly specify the projection matrix $\mathbf{P}$ to linearly project the data and compute the optimisation criterion of choice. Given the matrix $\mathbf{P} = [a_{ij}], i = 1, \ldots, d, j = 1, \ldots, k$ of size $d \times k$ the data is first cast into the new space

$$\mathbf{X}^{\mathrm{P}}_{N \times k} = \mathbf{X}_{N \times d} \mathbf{P}_{d \times k} \tag{2.9}$$

and then the relevance criterion is computed on this newly constructed data set $\mathbf{X}^{\mathrm{P}}_{N \times k}$. In this setting, scaling is a special case since it can be represented as a $d \times d$ matrix with the weights w_i on the main diagonal of the projection matrix, i.e.,

$$\mathbf{P}_{\mathrm{S}} = \begin{bmatrix} w_1 & 0 & \dots & 0 \\ 0 & w_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & w_d \end{bmatrix}_{d \times d} . \tag{2.10}$$

A good property of the projection method is the ability to linearly transform the data set to a lower dimensional space when the matrix $\mathbf{P}_{d \times k}$ has less columns than rows $k < d$. However, the number of parameters in $\mathbf{P}_{d \times k}$ is dk, and all have real values from $\mathbb{R}$, and thus the problem becomes even harder compared to the scaling problem with d parameters in a limited range. Furthermore, the correct value of k, the number of dimensions to project to, is an additional parameter that has to be optimized. The advantage is the manual choice of k, enabling full control of the dimensionality of the formed data set $\mathbf{X}^{\mathrm{P}}$.

2.4 Delta Test

Delta Test (DT) is used as a relevance criterion in many publications in this thesis. It is a noise variance estimator which tries to provide the information about the training error that can be achieved on the available training samples. If the functional dependence between the inputs x_i and the output y_i is given by Eq. (2.2), with the usual assumption of independent and identical distribution for the noise term, i.e., $\epsilon \sim \mathcal{N}(0, \sigma^2)$, then the Delta test returns an estimate of σ^2. This can be useful before building any model as it provides *a priori* estimate of the nonlinear correlation between the inputs and the output. It is a *nonparametric* noise estimator based on the nearest neighbour principle. The nearest neighbour of a point is defined as the unique point which minimizes a distance metric to that point. Distance metric is usually the Euclidean distance, but other metrics can also be used. It is based on the hypothesis coming from the continuity of the regression function. If two points x and x' are close in the input variable space, their corresponding outputs $f(\mathrm{x})$ and $f(\mathrm{x}')$ should be close in the output space. If this is not the case, one possible explanation is that this effect is due to the influence of the noise.

The use of the DT for variable selection is suggested in [13] with the idea of finding the subset of input variables such that the DT is minimised, that is, the subset of variables that represents the relationship in the most deterministic way.

DT is a special case of the Gamma test, another noise variance estimator suggested in [65]. The difference is that the Gamma test contains a single parameter p that controls the number of nearest neighbours needed to compute the estimate. DT is based only on the first neighbour of a sample in the input space, and thus removes this hyper-parameter completely providing fully nonparametric approach. For the proof of convergence of the DT to the true value of the noise variance see [14].

Let us denote the nearest neighbour of a point $\mathbf{x}_i \in \mathbb{R}^d$ as $\mathbf{x}_{NN(i)}$. The nearest neighbour formulation of the DT estimates the variance of the noise ϵ by

$$\mathrm{E}\left[\epsilon^2\right] \approx \frac{1}{2N}\sum_{i=1}^{N}(y_i - y_{NN(i)})^2\,, \tag{2.11}$$

where $y_{NN(i)}$ is the output of $\mathbf{x}_{NN(i)}$. For the variable selection problems, the goal is to *minimize* the value of the criterion (2.11) as this value represents the MSE that can be reached without overfitting. Thus, a lower value of the DT implies a better selection of variables [13].

2.4.1 Computation of Nearest Neighbours

Nearest neighbour search is an optimisation technique for finding the closest points in metric spaces. Specifically, given a set R of N reference points and a query point q (both the set and a point are in the same metric space V), the goal is to find the closest or nearest point $c \in R$ to q, i.e., $c = \mathrm{argmin}_j\{d(r_j, q), r_j \in R\}$ where $d(q, p)$ is the distance metric between two points $q, p \in V$. Usually, V is a d-dimensional space $\mathbb{R}^d$ and the distances are measured using Minkowski metrics. The search is also generalised for a set of query points Q, and the goal is to find either a nearest neighbour or k nearest neighbours to *all* query points in Q. In the DT computation, the sets R and Q are equal, and it is the data set $\mathbf{X}$ that is taken as both the reference and the query set. In this setting, when searching for the nearest neighbour of a sample $q = \mathbf{x}_i$, the search actually finds the two closest points c_1, c_2 and discards the closest point c_1 since that is the query point itself $\mathbf{x}_i$.

Naive Approach

To compute the DT value (2.11) for a certain solution S, the closest neighbour for each sample needs to be found in the input space. The simplest way, called the naive approach or brute force, is to find $NN(i)$ for each

sample independently of others. This method is computationally very demanding since the running time is of $O(dN^2)$ order. When coupled with the variable selection, this further introduces additional burden as certain previous calculations are discarded completely. If Euclidean metric is used to compute the nearest neighbours, i.e.,

$$\| \mathbf{x}_i - \mathbf{x}_j \|_2^2 = \sum_{l=1}^{d} (x_i^l - x_j^l)^2 \tag{2.12}$$

then the squared differences computed for one solution S_1 are lost even though they might be required for another solution S_2 if the two solutions have at least one common variable included $S_1 \cap S_2 \neq \emptyset$. One possibility is to store all the possible squared differences across all samples and all the features, resulting in a large matrix of size $N(N-1)/2 \times d$. For data sets of moderate size this approach is feasible on today's computer hardware, but for large data sets with $N > 1000$ even this method requires a lot of computational time.

Data Structures for Nearest Neighbours

To overcome the naive approach and its computational drawback, other methods have been proposed [66, 67] which use the data structures based on the decomposition of the multi-dimensional spaces. The main advantage of any decomposition based algorithm is the running time of the query searches, reducing the run time from $O(dN)$ to usually $O(c \log N)$ with c a factor depending on dimensionality d of data. This is a huge improvement for a query time, although certain amount of time and space are needed to construct the initial underlying data structure. Space requirements also enable more compact representation of the data and these vary depending on the algorithm being employed. One of the most well-known algorithms for the nearest neighbour search is the *kd-tree* [68]. For this algorithm and a reference set with N points and d attributes, the first phase that builds the tree requires $O(dN \log N)$ time and $O(N)$ space. For a new query point q, the expected computation of the query search takes $O(\log N)$ time. However, even this widely used decomposition suffers as the dimensionality increases as constant factors hidden in the asymptotic running time grow at least as fast as 2^d.

2.5 Contributions and Results

This section presents the contributions in the domain of variable selection coupled with the Delta test as the main relevance criterion.

2.5.1 Fixed Scaling (Publication II)

In many real world data sets, the number of samples is sometimes so large ($N > 10000$) that optimizing the scaling weights takes a considerable amount of time. When the nearest neighbour based methods are used this optimisation requires close to $O(dN^2)$ computations for a single iteration. One approach to solve this would simply be to randomly discard some portion of the samples in order to speed up the calculation time, but there is a risk of losing valuable data and there is no clear method to select important samples. Instead of removing the samples, a different strategy involves drastically reducing the number of variables by forcing most of the scaling weights to have zero value ($w_i = 0$). To achieve this goal, an additional constraint is added to the problem which requires that at most d_f scaling weights have non-zero values. Therefore, d_f variables are *fixed* to be included in the final scaling vector and the remaining $d - d_f$ weights are forced to zero, effectively changing the dimensionality of the data set.

For both scaling problems (the standard and with fixed d_f variables) any search algorithm can be easily modified to include the additional constraint that simply excludes any solution with excess variables. A different approach would be to consider this as a multi-objective (MO) optimization problem [60], where one objective is the main relevance criterion, and the other objective is the minimisation of the absolute difference between the number of non-zero scaling weights and the desired value d_f, i.e.,

$$\begin{aligned} F_1(\mathbf{w}) &= \text{relevance criterion with the scaled data set} \\ F_2(\mathbf{w}) &= \left| d_f - \left| \left\{ w^j \neq 0 \mid j = 1, \ldots, d \right\} \right| \right|. \end{aligned} \tag{2.13}$$

This approach is suggested in Publication II with the value d_f fixed to the half of the total number of variables in the data $d_f = d/2$. Table 2.1 shows the DT values obtained with the standard scaling approach and the version with the fixed number of variables and their computational times on several time series.

The underlying search algorithm is the Genetic Algorithm and the complete setup is described in the Publication II. Results presented in Table 2.1 are average values of the Delta test over 10 repeated runs of the al-

data set	DT		time (s)	
	standard	fixed	standard	fixed
Santa Fe	0.0075	0.0081	46.3	74.1
Poland electricity	0.0347	0.0379	83.2	102.5
Housing	0.055	0.066	35.7	50.1
Tecator	0.0102	0.0107	31.1	55.3

Table 2.1. Performance of the standard and the fixed scaling in terms of the DT values Eq. (2.11) and the execution times.

gorithm. Overall, having all variables included enables to explore wider solution space and to reach smaller DT values compared to the fixed scaling approach. On the other hand, DT values with half the number of variables provide reasonably good results which can benefit any modelling method that follows variable scaling as the dimensionality is greatly reduced. The running time is increased for the fixed scaling even though the nearest neighbour search is done in a reduced feature space. This is the downside of the multi-objective approach using the NSGA-II algorithm where it is necessary to sort which solutions should be kept for the next generation. This extra effort comes at the additional computational cost surpassing the execution time for the standard scaling method.

2.5.2 Combining Scaling and Projection (Publication II)

Consider the actual regression problem given by Eq. (2.2). One tries to build one or more models on the available data in order to predict the unseen samples. When it comes to the predictive power, the *model combining* or *model ensembling* approach provides much better accuracy than any model considered alone [69,70]. This issue is further discussed in Section 3.3. In the model combining approach, some form of model averaging is used to combine different behaviours of the models into a more powerful method.

Variable selection can be considered as another form of combating the issue between the model selection and the model combining. A single model in this strategy is a model $\mathcal{M}_1 = \mathcal{M}(S_1)$ trained on a solution S_1, that is, a subset of variables. A different model for the model combining is the same model $\mathcal{M}$ but trained on a different solution S_2, i.e., $\mathcal{M}_2 = \mathcal{M}(S_2)$. This way we obtain a set of models $M = \{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_L\}$ trained on L different solutions $\{S_1, S_2, \ldots, S_L\}$. The purpose is then to find the *model weights* to obtain a final model $\mathcal{M}_f \cong \sum_{i=1}^{L} w^i \mathcal{M}_i$ where $\cong$ indicates the

final model is an assembly of all other trained L models.

Similar principles can be applied to both variable scaling and variable projection. Instead of just building a single scaling solution, or a single projection solution, the problem can be attacked with *both* methods. This combination aims to shape the data and has the following form:

$$\mathbf{X}^{\mathrm{SP}}_{N\times(d+k)} = \left[\mathbf{X}^{\mathrm{S}}_{N\times d}, \mathbf{X}^{\mathrm{P}}_{N\times k}\right] \ , \tag{2.14}$$

where $\mathbf{X}^{\mathrm{S}}$ is the scaled version of $\mathbf{X}$ (Eq. (2.8)), $\mathbf{X}^{\mathrm{P}}$ is the projected version of $\mathbf{X}$ (Eq. (2.9)), and $\mathbf{X}^{\mathrm{SP}}$ is the new scaled and projected input matrix. The new matrix that needs to be optimized has the form:

$$\mathbf{P}_{\mathrm{SP}} = \underbrace{\begin{bmatrix} w_1 & 0 & \cdots & 0 & a_{11} & a_{12} & \cdots & a_{1k} \\ 0 & w_2 & \cdots & 0 & a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_d & a_{d1} & a_{d2} & \cdots & a_{dk} \end{bmatrix}}_{\mathbf{P}_{\mathrm{S}}\qquad\qquad\qquad\qquad\mathbf{P}_{\mathrm{P}}}{}_{d\times(d+k)} \ , \tag{2.15}$$

where $\mathbf{P}_{\mathrm{S}}$ is the same as in Eq. (2.10) and is responsible for scaling the data, while $\mathbf{P}_{\mathrm{P}}$ is the "classic" projection given by Eq. (2.9).

With the combination of scaling and projection, the optimisation stage should be able to reach a value F (relevance criterion) that is not worse than the value obtained using scaling or projection alone. This observation can be seen from the following two special cases. In the first special case, the projection columns are set to zero values, i.e., $\mathbf{P}_{\mathrm{P}} = \mathbf{0}$, and we have $\mathbf{X}^{\mathrm{SP}} = \left[\mathbf{X}^{\mathrm{S}}, \mathbf{0}_{N\times k}\right]$. This special case is just a scaling problem with the additional zero columns that do not influence the search process, but only increase computational time. The second special case is similar, with all the scaling weights set to zero $\mathbf{P}_{\mathrm{S}} = \mathbf{0}$, and we have the new data set $\mathbf{X}^{\mathrm{SP}} = \left[\mathbf{0}_{N\times d}, \mathbf{X}^{\mathrm{P}}\right]$, which is a pure projection problem with the extra computational cost. These two extreme cases suggest that by allowing both the scaling weights in $\mathbf{P}_{\mathrm{S}}$ and the elements in $\mathbf{P}_{\mathrm{P}}$ to have real values, it becomes possible to find solutions that are at least as good as the solutions for either the scaling or the projection problem.

Optimisation of the combined scaling and projected approach has a total of $d + dk = d(k + 1)$ parameters, compared to the pure scaling with d and the projection with dk parameters respectively. The difference between the combined method and the projection alone is only d parameters, which is also the difference between the two projection matrices of sizes $d\times k$ and

data set	DT		time (s)	
	DTSP-1	DTFSP-d/2-1	DTSP-1	DTFSP-d/2-1
Santa Fe	0.0056	0.0061	49.8	81.2
Poland electricity	0.028	0.03	84.3	113.5
Housing	0.046	0.047	34.2	69.3
Tecator	0.0029	0.0038	32.8	58.8

Table 2.2. Performance of the combined scaling and projection method in terms of the DT values Eq. (2.11) and the running times. DTSP-1 is the scaling and projection with one dimension, while DTFSP-d/2-1 is the scaling with the fixed number of variables (half the data set) plus the projection with one dimension.

$d \times (k + 1)$. Instead of projecting to more and more dimensions it might be fruitful to include scaling to replace that extra dimension. Projection is more difficult to optimise as it includes more local optima compared to the scaling approach. By including scaling instead of an extra projection dimension, the returned solution should in principle contain the good solutions from the pure scaling. The extra dimensions from the projection part are included to offer more flexibility compared to the scaling alone. On the other hand, in terms of the computational speed, the combined method takes much longer time since we are increasing the dimensionality from k dimensions in the projection to $k + d$ dimensions.

The combined approach can be extended to include the fixed version of scaling explained in Section 2.5.1 if that approach is necessary for the problem.

Table 2.2 shows the DT values with the same setup as in the previous section. Two methods are presented: 1) scaling and projection using only one extra dimension (labelled DTSP-1) and; 2) fixed scaling with half the variables in the data set and the projection using one extra dimension (labelled DTFSP-d/2-1).

The results in Table 2.2 are directly comparable to those given in Table 2.1. Out of all given methods, the scaling plus projection with one extra dimension produces the lowest DT values on all data sets. Including the fixed scaling in this approach has the similar effect as it did in the scaling method alone: the DT values are close to the version without it, while there is a significant increase in execution time. Having one extra projection dimension can help drastically when optimising the DT which can improve the predictive performance of the models in the subsequent stages.

One question that remains is whether one projection dimension is enough

since using more than one can further decrease the DT value. However, this introduces another hyper-parameter into the complete methodology which can prove to be prohibitive if some form of the validation procedure is used. Next section describes how to automatically choose the adequate projection dimension for this type of a problem.

2.5.3 Selecting Number of Projection Dimensions (Publication I)

For both the projection approach and the combined scaling plus projection method one needs to choose the appropriate value for k in Eq. (2.9) and Eq. (2.15). One method to automatically choose the suitable value for k comes from the following observation. Consider the case when the projection matrix $\mathbf{P}_{d\times k}$ has one column $k = 1$, and suppose that by optimizing a relevance criterion we can obtain the value $F(\mathbf{P}_{d\times 1})$. Same $F(\mathbf{P}_{d\times 1})$ value can be obtained with $k = 2$ by setting the second column of the matrix $\mathbf{P}_{d\times 2}$ to zero values. This setting does not have any influence during the optimisation process, resulting in the same optimisation problem as with the matrix $\mathbf{P}_{d\times 1}$.

Since $\mathbf{P}_{d\times 2}$ contains real values, optimising $\mathbf{P}_{d\times 2}$ should be able to reach the value $F(\mathbf{P}_{d\times 2})$ that is at least as good as $F(\mathbf{P}_{d\times 1})$. However, adding new d parameters to $\mathbf{P}_{d\times 1}$ increases the complexity of the problem, adds new local optima and the optimisation of $\mathbf{P}_{d\times 2}$ becomes more challenging. This complexity is manifested through the value $F(\mathbf{P}_{d\times 2})$, which can be worse than $F(\mathbf{P}_{d\times 1})$, if both optimisation problems are given the same amount of resources.

The same reasoning applies to the higher values of k. Thus, as k is increasing, the value of the relevance criterion F should always improve. In practice, huge number of parameters prevents $\mathbf{P}$ matrices with large k to reach the same results of those cases with the lower k. When optimising F as a projection problem, there will be a value of $k = k_p$ after which the search procedure is unable to return lower F values. Thus, we can conclude that the matrix $\mathbf{P}_{d\times k_p}$ is our best estimate and considering $k > k_p$ values means wasting resources. Assuming the same amount of resources is spent at each step, at some point of the optimisation the criterion F will no longer improve and the "optimal" value for k_p can be established. On the other hand, different amounts of effort at each step would quickly lead to high waiting times even for the smaller values of k. Moreover, the optimality of each solution $\mathbf{P}_{d\times k}$ cannot be guaranteed, and it is difficult

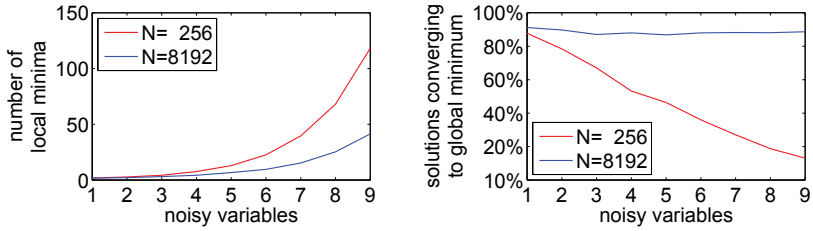

Figure 2.2. Number of local minima and the percentage of solutions from which the FBS converges to global minimum.

to justify spending more resources on higher values for k.

The suggested approach is combined with the two fast models Optimally Pruned Extreme Learning Machine (Section 3.3.2) and Optimally Pruned k-Nearest Neighbours for two tasks: regular regression for one specific financial data set and time series prediction.

2.5.4 Optimisation in Large Sample Data (Publication IV)

An interesting phenomenon in the variable selection with the Delta test is the formation of the solution space when there is a large number of samples. As the number of samples increases, the DT should return a good approximation of the noise variance. As suggested in [13], the DT can be used as a variable selection criterion to discriminate between the important variables and the irrelevant ones. The exhaustive search is used to find the correct subset of variables that fully determine the target variable. In the case of several tens of variables it is not possible to use the exhaustive search to find out the optimal set of variables. On the other hand, the transition between the solutions (considering the neighbourhood structure described in Section 2.2.3) enables the use of the simple Forward-Backward Search to find the global minimum.

This comes from the following observation outlined in Figure 2.2 showing a synthetic data set with 3 relevant variables and several irrelevant ones. There are two cases of interest: small sample size ($N = 256$) and large sample size ($N = 8192$). As the number of samples increases for the same number of variables, the number of local minima decreases. The other aspect is that from the most solutions a simple FBS procedure converges to the global minimum.

To reach the global minimum, several or possibly about 20 starting solutions must be generated. During these restarts, a lot of information about the landscape is ignored. This information can be used to construct

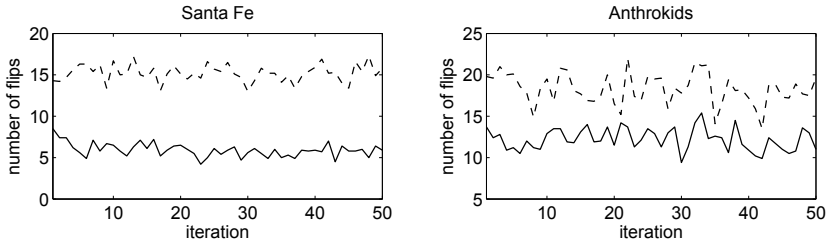

Figure 2.3. Number of transitions for two data sets as a function of number of iterations.

more promising starting solutions such that FBS requires less steps to converge to the local minimum. The restarting idea is of interest in many domains [71–73].

Long-term memory structures can be used to keep track of different aspects of the search process, which in turn are used to generate new starting positions. The most commonly used aspects of variables are *constancy* and *strong determination*. The former notion indicates how many times a variable is present in the best solutions found, while the latter informs how much the objective function is influenced by a change of that specific variable. Other information can be used as well, but for variable selection it suffices to use only those two. Both of these notions are used in the first stage of the procedure – the construction phase. Each piece of information contributes to the energy of a variable $E(i)$, $i = 1, \ldots, d$. The construction is done by adding a single variable at a time, usually in a probabilistic setting. The probabilities are obtained from the energies in the usual way, i.e., $p(i) = E(i)/\sum_{i=1}^{d} E(i)$. After the construction phase, the greedy algorithm (FBS) is used to find the local optimum. The construction phase and the convergence (descent phase) constitute one iteration of the approach. This method provides quicker convergence times as the required number of transitions is reduced compared to the purely random starting positions. Figure 2.3 shows this effect for two data sets, Santa Fe time series with 12 time lags as the input regressor and Anthrokids. For Santa Fe, the restarting strategy reduces the number of transitions to about a third, while for Anthrokids the improvement is less substantial but better than the random during all 50 iterations.

2.5.5 Parallel Implementations (Publications VII and VIII)

Variable selection with the Delta test has one huge drawback – the computation of the DT itself, which requires huge amount of time since it is based on the nearest neighbour search. This issue is tied with both the

number of samples and the number of variables. The computational requirement scales linearly with the input dimension, while the number of samples has quadratic influence on the running time. Several solutions are possible to alleviate this problem: 1) better data structures for the nearest neighbour searches; 2) approximate nearest neighbour computation to reduce the execution time at the expense of accuracy; and 3) taking advantage of the high computing architectures available today.

In the domain of parallel GAs there is a trade-off between the execution time and the number of constructed generations. The performance of the algorithms can be measured by keeping one of these values fixed and monitoring the other. For example, the number of generations can be kept fixed, while measuring the execution time among the tested algorithms. The algorithm with the fastest execution time is deemed superior over the other candidates. On the other hand, the maximum allowed computation time can be set in advance, and in this setting, the algorithm with more evaluated individuals/generations is favourable over the other variants. Having more evaluated generations should in theory explore a wider area of the search space, and thus, return a better solution. The reason for keeping the execution time fixed is the applicability of this approach for the time critical tasks.

Publication VIII showcases two possible approaches. One deals with a network of homogeneous computers while the other is adapted for a situation of heterogeneous computers with the GPU capabilities. Both approaches use the GA as the main search algorithm.

Homogeneous cluster. The first approach is the case when several nodes (computers) have more or less equal processing power. In this situation, a simple master/slave approach is adopted where one node is devoted to the computing operators of the GA (sequential part of the GA), while all the nodes evaluate the individuals in the population. Since transferring of individuals is a rather expensive step, the algorithm is made so that all the nodes have *one population* at their disposal. When the communication occurs, the master slave only needs to send the random number seed in order for the slaves to find which individuals they should focus on. Another important step is the computation of the *distance tensor* which stores all the pairwise squared differences between the samples for all the features, i.e., $(x_i^l - x_j^l)^2$, $i, j = 1, \ldots, N$, $l = 1, \ldots, d$. This computation is distributed among the nodes before the execution of the actual optimisation of the DT. This way, when the actual squared distance is required for a subset of

data set	population	RCGA	pRCGA (np=2)	pRCGA (np=8)
	50	0.01278	0.01269	0.01347
Anthrokids	100	0.01351	0.01266	0.01115
	150	0.01475	0.01318	0.01105
	50	0.01422	0.01452	0.01403
ESTSP'07	100	0.01457	0.01419	0.01393
	150	0.01464	0.01429	0.0141

Table 2.3. Delta test values Eq. (2.11) for variable selection for the sequential GA (RCGA) and the parallel version (pRCGA) with different number of processors (np).

variables I among two samples, only the summation over those I tensor entries is needed.

Table 2.3 summarises this parallel implementation with different population sizes and the number of processors. The execution time is fixed to 600 seconds and the problem is pure variable selection. Two data sets are considered: Anthrokids and a time series from the ESTSP'07 conference competition.

As expected, with increased population size, as well as with more processing power, it is possible to reach smaller DT values than with a sequential GA executed on a single CPU.

Heterogeneous cluster. The second approach suggested is when the cluster of machines contains the GPU processing capabilities. Since the memory capacity of the GPUs is much smaller than the standard system memory, the approach with the precomputed distance tensor is not viable anymore. In this situation, using the standard approach for finding the nearest neighbours is still much faster using a GPU as has been shown in [74]. As the method is developed for the heterogeneous architectures, the natural way is to consider each node in this network as a separate process which leads to the *island model*. Each CPU/GPU pair is considered to be one island and responsible for one specific population of the individuals. Since several populations are being evolved at the same time, the question is how information should flow between the populations. A simple fixed migration scheme where the best individuals are exchanged between the populations is adopted. This migration is done after a predefined number of generations has passed.

Since the migration scheme is fixed, there is a possibility that the faster machines have to wait for the slower ones to finish their execution. This can be solved by adjusting the number of individuals each cluster needs to

evaluate. This is done dynamically depending on the performance of the fastest machine Γ_f. The populations are adjusted as follows: if Π_f is the total number of individuals for the fastest machine, then each machine Γ_i gets the following amount of individuals – $\Pi_f \cdot \text{time}(\Gamma_i)/\text{time}(\Gamma_f)$. Thus, each machine gets a fraction of individuals corresponding to the ratio of the execution time between the machine in question and the fastest one. This approach is suitable for situations where there are couple of tens of thousand of samples and the execution time is of concern.

3. Time Series Prediction

This chapter explains the basics of time series prediction, from the starting point to the advanced methods needed to solve one of the problems in this domain. The prevalent and most important task is how to predict the future values for a time series based solely on the available past samples. This issue is addressed with several novel methods: one is designed to tackle this issue for a small number of samples, while the other contains implicit ranking of the *time lags* or previous values most relevant for the accurate estimation of the future values.

3.1 Basics of Time Series Prediction

Time series prediction is needed in many fields of science and industry. It is related to the single phenomenon measured over the course of time. These subsequent measurements (samples) are indexed using integers which results in a series of values – z_t, $t = 1, 2, \ldots$. The true time delay (interval) between between these samples can be anything, for example, one hour or 10 years, depending on the problem setting. The time indices t usually form a discrete set ranging from 1 to T, with z_1 being the first measurement and z_T the last. For further simplification, we use a vector notation $\mathbf{z}$ to indicate the whole series $\mathbf{z} = [z_1, z_2, \ldots, z_T]$. Almost all real-world time series can be considered *stochastic* time series, that is, there is an underlying process or mechanism that generates this series with an added noise term. This assumption is very useful and also follows broader class of regression problems represented by Eq. (2.2).

Several problems can be associated with the time series prediction [75, 76], but the most difficult and of great practical importance is the prediction of future values $\hat{z}_{T+1}$. In this setting, we are interested to use previous values of z to predict the next value in the sequence. This can be

modelled as

$$\hat{z}_{T+1} = f(z_T, z_{T-1}, \ldots, z_{T-h+1}, \ldots) + \epsilon\,. \tag{3.1}$$

Usually, only certain amount of previous values is taken into consideration, for example, h previous ones. Term h is called the time lag and is closely related to the problem of model selection. In the ideal setting, the subsequent measurements are taken at an equidistant sampling interval, that is, they are equispaced appearing after a constant time frame, for example, each day in a year, or every hour. This is possible if the measuring instruments are precise and the acquisition is sufficiently easy. In the environmental modelling community, where most of the data come from either marine or riverine systems, such an approach is impossible. Most of the time these series exhibit highly irregular sampling, where the time interval between the successive values can be from couple of days to several weeks. These and other issues related to the sampling in environmental data are further discussed in Chapter 4.

Time series analysis is one approach to understand the underlying stochastic mechanism. One of the "easier" aspects of the time series is the seasonality. In that case, the time series has more or less recurring pattern over some period of time. For example, temperature in a local area has a yearly pattern which depends on the sampling interval. If the measurements are taken every day, this pattern is shown over a range of 365 values. Another aspect is the global or the general trend of the series. A simple example would be the total amount of the world wealth which can be considered to be always increasing. This increasing is the global trend of the series, and in certain situations, that is, certain time series, do not have an evident global trend.

For over several decades, time series forecasting has only been focused on providing an answer to the *single* future value $\hat{z}_{T+1}$. This approach is called one-step ahead or short-term prediction. If the practitioner is interested in more general trend in the future, the *same* model that was build to predict $\hat{z}_{T+1}$ is used to predict the values at time steps $T+2$ and beyond. The difference here is that the value z_{T+1} is unknown, and the only available information at this modelling stage is the predicted value $\hat{z}_{T+1}$. This is then used to obtain $\hat{z}_{T+2}$, that is, the same model f is used where the series $\mathbf{z}$ is extended to include $\hat{z}_{T+1}$, i.e., $\mathbf{z}' = [z_1, z_2, \ldots, z_T, \hat{z}_{T+1}]$, and the prediction is then done in the following way

$$\begin{aligned}
\hat{z}_{T+1} &= f(z_T, z_{T-1}, \ldots, z_{T-h+1}) \\
\hat{z}_{T+2} &= f(\hat{z}_{T+1}, z_T, z_{T-1}, \ldots, z_{T-h+2}) \\
\hat{z}_{T+3} &= f(\hat{z}_{T+2}, \hat{z}_{T+1}, z_T, \ldots, z_{T-h+3}) \\
&\vdots
\end{aligned} \tag{3.2}$$

where h is taken to be the maximum number of previous lags used for prediction. This approach is known as the *recursive* method of time series prediction since the same model is used repeatedly for all the successive steps. The strategy is also known as iterative, one-step-ahead, or continuous prediction strategy.

In the case when several tens or even hundreds of values need to be predicted, we are talking about *long-term* prediction. The difficulty in long-term prediction using a single model is the accumulation of errors and uncertainties which makes the estimated values highly unreliable. The solution that has been proposed is the use of the *direct* strategy [77]. In this setting, for each future value $\hat{z}_{T+k}$, a *separate* model f_k is build with the index k indicating the time step the model is built to predict. This gives the following representation

$$\begin{aligned}
\hat{z}_{T+1} &= f_1(z_T, z_{T-1}, \ldots, z_{T-h+1}) \\
\hat{z}_{T+2} &= f_2(z_T, z_{T-1}, \ldots, z_{T-h+1}) \\
\hat{z}_{T+3} &= f_3(z_T, z_{T-1}, \ldots, z_{T-h+1}) \\
&\vdots
\end{aligned} \tag{3.3}$$

The main advantage of this approach is that the prediction is done exclusively on *known or available* data, and the modelling of the relationship between the lags and the required time step is captured by a model specifically trained for this task. In practice this provides more stable and accurate predictions [77]. The disadvantage is the time requirement, since each model has to be trained separately, and depending on the choice of the model and required prediction steps k, the training stage can take a considerable amount of time. The only case when the recursive strategy provides better performance is when the time series is close to linear in which case the linearity is captured by the first model f_1 or just f.

Another aspect worth mentioning is the case when time series $\mathbf{z}$ is accompanied by the external or exogenous series $\mathbf{z}^k$. The presence of the

extra information can be incorporated as part of the input to the model f. This is especially useful when the number of samples is low, and any extra addition is valuable for accurate predictions. This issue is discussed more in Chapter 4 with the application to the benthic data from the North Sea in the form of multi-factor analysis.

In order to train the model, a suitable data set is required where the training data consists of known input-output pairs. With h previous values taken to be the input, a *regressor* is build by sliding a window of length h over the complete series, and registering a range of h consecutive values as a sample for the model. Given the series with T values, the i-th sample in the regressor has the following values

$$(\mathbf{x}_i; y_i) = (z_i, z_{i+1}, \ldots, z_{i+h-1}; z_{i+h}), \quad 1 \leq i \leq T - h. \tag{3.4}$$

Once the data set has been formed, *any* kind of model can be used in a supervised fashion. In this approach, the temporal dependency is broken down into the time lags z_t, z_{t-1}, ..., z_{t-h+1}. The lags now act as features or variables making the methods from the supervised domain readily available. Variable selection procedures discussed in Chapter 2 can easily be applied to this new setting. A solution obtained from a variable selection method provides an insight into the most important time lags for a specific time series. For example, for a time series representing daily electricity consumption, a reasonable lag to select would be the same day of the previous week, that is, a value z_{t-7}. Another advantage of the representation given by Eq. (3.4) is that newly developed supervised methods can be directly applied to the time series data. The one downside to this approach is the lack of a sound theoretical basis.

A different approach would be to consider prediction as a missing value problem [78, 79] where many future values are considered to be "missing" instead of being unknown. With this strategy, several future values can be predicted at the same time with a single model that is taking into account the dynamics of the time series.

3.2 Basics of Linear Models

In this section, we focus on a class of linear models based on the Box-Jenkins methodology [75]. These models include auto-regressive (AR), integrated (I), moving average (MA) and their combination auto-regressive moving average (ARMA), the auto-regressive integrated moving average

(ARIMA), as well as seasonal version (SARIMA). The underlying assumption behind these models is a linear relationship between measurements themselves $\mathbf{z}$ and a linear relationship with random independent shocks or noise ϵ. Although the assumption of linearity might be too restrictive in most cases, these models can be useful as a baseline and are able to describe homogeneous non-stationary behaviour.

We only give brief descriptions of these models, while their characteristics, like variance, autocorrelation functions, spectra, stationarity conditions for their parameters are not discussed. This information can be found in the literature [75, 80].

3.2.1 Linear Filter Models

Time series are regarded as the stochastic processes. The assumption is that there is an underlying mechanism generating the series, but the nature of this process is random due to the unknown influences which are then considered as noise. One way to model this mechanism is to assume that successive values are highly dependent and can be regarded as generated from a series of independent shocks a_t. These shocks are coming from a fixed probability distribution, usually Gaussian with zero mean and constant variance σ^2. This is represented as

$$z_t = \mu + a_t + \eta_1 a_{t-1} + \eta_2 a_{t-2} + \cdots \tag{3.5}$$

and the result is a weighted sum around the level or mean of the process μ. This is simply a linear transformation of shocks, or a linear filter model. The model representation allows a range of patterns of dependence among the values of the process z_t expressed in terms of the unobservable random alterations a_t. The sequence of coefficients $\eta_1, \eta_2, \ldots$, can be either finite and infinite. Depending on the properties of this sequence, the process z_t can be stationary or non-stationary. Stationary process is obtained when the sequence is finite, or infinite and absolutely summable $\sum_{i=1}^{\infty} |\eta_i| < \infty$. Otherwise, the process is non-stationary.

3.2.2 Autoregressive Process

A useful model to represent the time series is an *autoregressive process* (AR). In this model, the current value z_t is expressed as a finite, linear combination of the previous values of the process plus a random shock

$$z_t = \gamma_1 z_{t-1} + \gamma_2 z_{t-2} + \cdots + \gamma_P z_{t-P} + a_t \,. \tag{3.6}$$

This representation is the autoregressive model of order P. The coefficients $\gamma_1, \ldots, \gamma_P$ are constants while a_t is the noise term driving the process. Sometimes a_t is called the innovation term and without it, the AR process is completely deterministic. Sometimes, the modelling is performed on the differences from the mean level μ, $\tilde{z}_k = z_k - \mu$ for all values of k and the Eq. (3.6) is reformulated to include the modified values $\tilde{z}_k$ instead of original values z_k. For clarity, we assume that $\mu = 0$ and we omit it from the equations.

Autoregressive process can be transformed into a linear filter model by substituting each z_k with its own representation. For example, with AR process of order $P = 1$, $z_t = \gamma_1 z_{t-1} + a_t$, after m substitutions of $z_{t-j} = \gamma_1 z_{t-j-1} + a_{t-j}$ we have

$$z_t = \gamma_1^{m+1} z_{t-m-1} + a_t + \gamma_1 a_{t-1} + \cdots + \gamma_1^m a_{t-m} \tag{3.7}$$

which in the limit $m \to \infty$ leads to a convergent infinite series representation $z_t = \sum_{j=0}^{\infty} \gamma_1^j a_{t-j}$ provided that $|\gamma_1| < 1$. Autoregressive processes can be both stationary and non-stationary, and this depends on the coefficients γ_j. Since the AR model can be represented as the infinite series of shocks, the condition for stationarity is that the coefficients γ_j are absolutely summable as already mentioned.

The total number of parameters in the AR model is $P + 2$, with P the number of γ_j coefficients and the remaining two parameters are the noise variance of the added shock term a, plus the mean μ of the time series. Sometimes the mean is assumed to be zero (in order to simplify equations), in which case there is one less parameter to estimate. Autoregressive models are popular because their coefficients can be solved from a set of linear equations and they are good approximators for some practical time series.

3.2.3 Moving Average Models

Another simple model of great importance is the *moving average* (MA) model, where the dependence is made on the Q previous *shocks* themselves

$$z_t = a_t - \eta_1 a_{t-1} - \eta_2 a_{t-2} - \cdots - \eta_Q a_{t-Q} \,. \tag{3.8}$$

The total number of parameters for this model is $Q+1$ if we assume zero mean time series.

Both AR and MA models can be more economically described with the *forward* and *backward* operators that are common in the literature, but we are keeping the notation simple for the brevity and connections with the previous chapter. Solving the coefficients of the MA model requires nonlinear optimisation.

3.2.4 Autoregressive Moving Average

The linear filter model given by Eq. (3.5) where the process is a sum of random shocks can have an alternative form as a sum of previous values of the process z_{t-j} [75] under suitable conditions. As a special case, MA(1) process can be represented as an infinite sum of previous z_{t-j}, $j = 1, 2, \ldots$ values, that is, an infinite autoregressive process. Similarly, AR(1) does not have a parsimonious representation as a moving average model. In order to force parsimony and to achieve greater flexibility in fitting of the actual time series, it is sometimes advantageous to include both the autoregressive and the moving average terms in the model. This leads to the mixed autoregressive moving average (ARMA) model:

$$z_t = \gamma_1 z_{t-1} + \gamma_2 z_{t-2} + \cdots + \beta_P z_{t-P} + a_t - \eta_1 a_{t-1} - \eta_2 a_{t-2} - \cdots - \eta_Q a_{t-Q} \quad (3.9)$$

The model contains $P + Q + 1$ unknown parameters that are estimated from the data. This model is also possible to represent in the form of a linear filter given by Eq. (3.5). In practice, most occurring stationary time series can be obtained with the three mentioned models: autoregressive, moving average, or mixed model, where the lags P and Q are not greater than 2.

3.2.5 Autoregressive Integrated Moving Average

The majority of the real-world time series encountered in the industry exhibit non-stationary behaviour and do not have a fixed mean μ. Even in those cases, the broad behaviour of the whole series may be similar if differences in the mean μ are allowed for. Such behaviour can be represented with a more powerful model – autoregressive integrated moving average (ARIMA) process of order (P, D, Q), where the D indicates the D-th difference in the series, i.e., $\bar{z}_t = z_t - z_{t-D}$, and this modified series is modelled with an ARMA(P, Q). When $D = 0$ the model is the standard ARMA(P, Q)

describing the stationary process. This model is described as follows

$$\bar{z}_t = \gamma_1 \bar{z}_{t-1} + \gamma_2 \bar{z}_{t-2} + \cdots + \gamma_P \hat{z}_{t-P} + a_t - \eta_1 a_{t-1} - \eta_2 a_{t-2} - \cdots - \eta_Q a_{t-Q} \,. \quad (3.10)$$

3.3 Nonlinear Approaches

In both machine learning and statistical community, many nonlinear models have been developed over the years. These vary depending on the task, from function approximation, density estimation, clustering, visualisation and dimensionality reduction. In all the approaches, the idea is to build a single model that will capture the underlying mechanism generating the data.

3.3.1 Neural Networks

Neural networks comprise of different types of models, such as feedforward networks, recurrent networks, networks for vector quantisation and many others [4, 64, 81]. The basic idea has biological inspirations as early as 1940's with the works of McCulloch and Pitts [82] trying to provide a mathematical representation of information processing in biological systems. The most dominant and widely used model is the *multilayer perceptron* (MLP) [4] developed from the early perceptron [83] model. The model consists of at least one *hidden layer* in addition to the input layer coming from the actual data. Each layer contains neurons that perform nonlinear transformations on their respective inputs. These transformations are the key concept of neural networks and they provide a framework that has the universal approximation capabilities [84, 85].

In this dissertation, we are interested in the feedforward networks and the structure of such networks is depicted in Figures 3.1 and 3.2. Figure 3.1 shows the layout a of single neuron which constitutes the basic processing unit in neural networks. Figure 3.2 presents a general structure with two hidden layers and each hidden layer comprises of the neurons depicted in Figure 3.1. The input variables x are the first layer of the network, simply called the input layer. These inputs are then fed to the next layer by multiplying them with the weights $\boldsymbol{\beta}_m^{(1)} = [\beta_{m1}^{(1)}, \ldots, \beta_{md}^{(1)}]$ to each neuron $g_m^{(1)}$, $m = 1, \ldots, M_1$, where M_1 is the number of neurons in the first layer. Weight $\beta_{mi}^{(1)}$, $i = 1, \ldots, d$ is associated with the synaptic connection between the i-th input variable and the m-th neuron in the hidden layer.

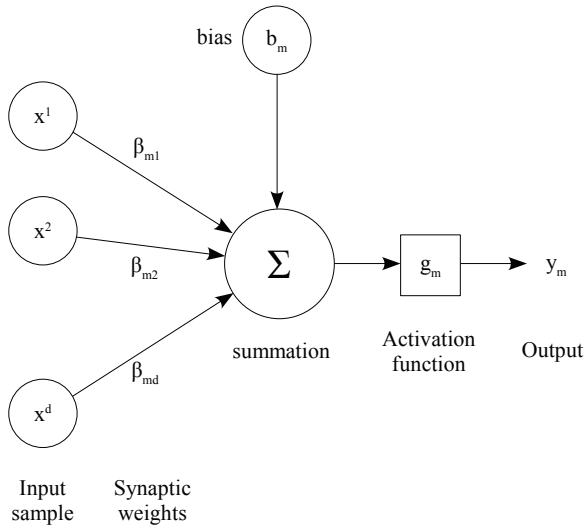

Figure 3.1. Structure of a neuron (assumed to be m-th neuron in the layer). The neuron processes information by multiplying the inputs x^i with the synaptic weights β_{ki}, after which the summed activation level $\sum_{i=1}^{d} \beta_{ki} x^i$ is put through a nonlinear transformation $g_m(\cdot)$ to produce the output y_m.

Considering a single sample $\mathbf{x}_i$, it is transformed by the first layer in two steps:

1. Computing activations for each neuron in the hidden layer

$$A_m(\mathbf{x}_i) = \sum_{j=1}^{d} \beta_{mj}^{(1)} x_i^j + b_m, \quad 1 \leq m \leq M_1. \tag{3.11}$$

2. Transforming the activations using a nonlinear function

$$g_m^{(1)}(\mathbf{x}_i) = g_m^{(1)}(A_m(\mathbf{x}_i)), \quad 1 \leq m \leq M_1. \tag{3.12}$$

The term $g_m^{(1)}$ is used to indicate two things for clarity: the neuron m in the *first hidden layer* and the activation function (transformation) for the same neuron. The bias of that m-th neurons is defined by b_m, $m = 1, \ldots, M_1$.

If all the samples are processed in the same manner, the first layer produces transformed data, which can be written in more compact matrix form as

$$\mathbf{H}_1 = \begin{bmatrix} g_1^{(1)}(\boldsymbol{\beta}_1^{(1)} \mathbf{x}_1^{\mathrm{T}} + b_1) & \cdots & g_{M_1}^{(1)}(\boldsymbol{\beta}_{M_1}^{(1)} \mathbf{x}_1^{\mathrm{T}} + b_{M_1}) \\ \vdots & \ddots & \vdots \\ g_1^{(1)}(\boldsymbol{\beta}_1^{(1)} \mathbf{x}_N^{\mathrm{T}} + b_1) & \cdots & g_{M_1}^{(1)}(\boldsymbol{\beta}_{M_1}^{(1)} \mathbf{x}_N^{\mathrm{T}} + b_{M_1}) \end{bmatrix}. \tag{3.13}$$

The size of $\mathbf{H}_1$ matrix is $N \times M_1$ with N being the number of samples.

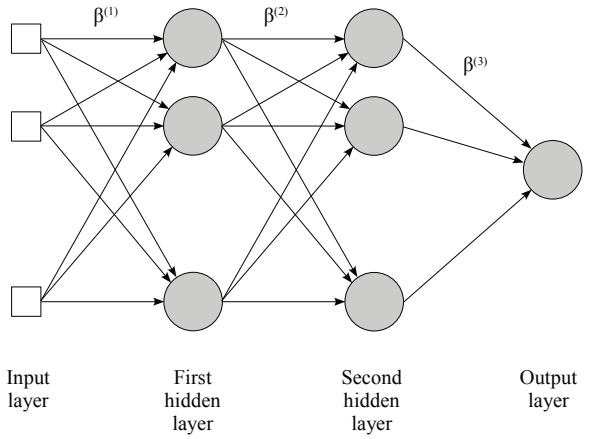

Figure 3.2. Feedforward neural network with two hidden layers. In this structure, three levels of synaptic weights are present – $\boldsymbol{\beta}^{(1)}$ between the input and the first hidden layer, $\boldsymbol{\beta}^{(2)}$ between two hidden layers and $\boldsymbol{\beta}^{(3)}$ between the second hidden layer and the output layer.

The usual choices for the activation functions g_m are the logistic sigmoid and the hyperbolic tangent $tanh$

$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}}, \qquad tanh(x) = \frac{e^{2x-1}}{e^{2x+1}}\,. \tag{3.14}$$

When the network has more than one hidden layer, the transformation process continues by inputting newly acquired $\mathbf{H}_1$ to the next layer with the weights $\boldsymbol{\beta}^{(2)}$ to obtain $\mathbf{H}_2$ and so on. The alternative way that is more commonly used in the literature is to process one sample through the complete network with the L hidden layers to the obtain output $F(\mathbf{x}_i;\boldsymbol{\beta})$, that is,

$$\mathbf{x}_i \overset{\boldsymbol{\beta}^{(1)}}{\rightarrow} \begin{bmatrix} g_1^{(1)}(\mathbf{x}_i) \\ \vdots \\ g_{M_1}^{(1)}(\mathbf{x}_i) \end{bmatrix}^{\mathrm{T}} \overset{\boldsymbol{\beta}^{(2)}}{\rightarrow} \begin{bmatrix} g_1^{(2)}(\mathbf{x}_i) \\ \vdots \\ g_{M_2}^{(2)}(\mathbf{x}_i) \end{bmatrix}^{\mathrm{T}} \overset{\boldsymbol{\beta}^{(3)}}{\rightarrow} \ldots \overset{\boldsymbol{\beta}^{(L)}}{\rightarrow} F(\mathbf{x}_i;\boldsymbol{\beta}) = \hat{y}_i\,. \tag{3.15}$$

Here we use the matrix notation given by Eq. (3.13) since it will provide a convenient representation for a specific type of a network discussed in Section 3.3.2.

In the case of a single hidden layer networks, the outputs from the hidden layer (matrix $\mathbf{H}_1$) propagate to the output layer, this time with the hidden layer output weights $\boldsymbol{\beta}^{(2)}$ which can be written in the matrix form $\mathbf{H}_1\boldsymbol{\beta}^{(2)}$. These are the input values for the output layer neuron(s). The output layer can be linear or nonlinear depending on the problem at hand or the network structure chosen. In the case of multi-output data, the

output layer contains more than one neuron y, but in this thesis we are only concerned with the univariate data sets. Given the structure of the feedforward networks, they simply perform a nonlinear function from a set of input variables to a set of output variables controlled by a vector of adjustable parameters $\boldsymbol{\beta}$. In these types of networks, the information *flows* from the input layer to the output layer, and this process is called *forward propagation*. Learning these models requires updating the vector of parameters $\boldsymbol{\beta}$ which is done by propagating information *backwards* from the output layer to the input layer.

The complete flow of information can be generalised to more than one hidden layer, but a from theoretical point of view, a neural network with one hidden layer and a linear output can approximate any continuous function on a compact domain to a desired level of accuracy [84, 85]. This property is noted the as universal approximation, and enables neural networks to represent a wide range of functions, although a good approximation may require a large number of neurons in the single hidden layer. In recent years, there has been a lot of development in deep learning, with structures comprising of several layers with a large number of neurons and interconnecting weights and algorithms for their efficient learning [5, 6]. In this dissertation, all the networks are with a single hidden layer since training these is much quicker and easier, and as we shall see in Section 3.3.2, there is an interesting type of the network with incredibly fast training stage, called Extreme Learning Machine.

Major advantages of the neural networks are their good generalisation property, applicability to a wide range of regression and classification problems. The downside is that such models are a black-box type of models, and interpretation of the adjustable parameters is extremely difficult.

Training. For regression problems, the training stage involves minimising the error function, usually sum of squared errors across the entire data, with respect to the adjustable parameters $\boldsymbol{\beta}$, i.e.,

$$E(\boldsymbol{\beta}) = \sum_{i=1}^{N} \|F(\mathbf{x}_i; \boldsymbol{\beta}) - y_i\|^2 \tag{3.16}$$

with $F(\mathbf{x}_i; \boldsymbol{\beta})$ being the output of the network for sample $\mathbf{x}_i$ with weights $\boldsymbol{\beta} = \{\boldsymbol{\beta}^{(1)}, \boldsymbol{\beta}^{(2)}\}$. This minimisation is highly nonlinear and there are many local minima, so the optimisation technique is not guaranteed to converge to the global minimum. Most of the times, this does not pose a problem provided that several training cases starting from different ini-

tial values are performed, and the one with the best generalisation performance is chosen as the final model. Since no analytical solution exists to find the minimum, iterative procedures are used where the parameters $\boldsymbol{\beta}$ are updated at every iteration. Most techniques start at some initial point of the parameters $\boldsymbol{\beta}^0$ and then move through the parameter space by applying the update rule of the form $\boldsymbol{\beta}^{\tau+1} = \boldsymbol{\beta}^{\tau} + \Delta\boldsymbol{\beta}^{\tau}$. The update rules usually employ information of the gradient of the error function. Among these methods, the stochastic gradient descent is the most well-know technique. In this approach, the update is performed by adjusting the parameters in the direction of the negative gradient, that is, $\boldsymbol{\beta}^{\tau+1} = \boldsymbol{\beta}^{\tau} - \eta\nabla E(\boldsymbol{\beta}^{\tau})$ where η is the learning rate. The updates are performed for a single data point at a time or some portion of the data instead of the complete data set. Thus, the gradient is replaced by a coarse instantaneous value, but when performing many small updates the algorithm proceeds roughly to the direction of the true gradient. In order to update the weights, the derivatives $\partial E/\partial\beta_{ij}^{(k)}$ are required, and an efficient method to compute these derivatives is done with the *error backpropagation* [86] where the information of the errors is send backwards through the network.

The following five steps summarise the error-backpropagation algorithm:

1. Initialize all the weights of the network to small random values.
2. Select an input vector from the available training set and calculate the output of the network for it.
3. Compare this output with the corresponding desired response, and calculate all the local errors for the output layer and for the hidden layers.
4. Update the weights accordingly based on the local errors.
5. Repeat steps 2 through 4 until the backpropagation algorithm has converged to a predetermined level of accuracy.

There exist more efficient methods, such as Levenberg-Marquardt and conjugate gradient algorithm [4, 81].

Complexity. Besides the adjustable parameters $\boldsymbol{\beta}$ optimised in the training phase, the number of neurons in the hidden layer M is a free parameter that needs fine tuning. This number must be chosen carefully to balance between under-fitting and over-fitting which can be done with the validation methods. Other methods involve regularising the network weights (weight-decay) [87] or an early stopping approach [88].

3.3.2 Extreme Learning Machine

Different from the standard error-propagating algorithms, a fairly new neural network learning method has emerged in the form of Extreme Learning Machine (ELM) [89]. The novelty is the complete removal of any iterative training steps by *random* initialisation of the hidden layer input weights $\boldsymbol{\beta}^{(1)}$ and biases. This random initialisation removes the weights $\boldsymbol{\beta}^{(1)}$ in the first layer from the iterative updates in the backpropagation algorithm. Since these weights remain fixed during the learning stage, we can propagate all the samples through the first layer to obtain the transformed data (Eq. (3.13)). ELM also contains only one single hidden layer and with the assumed linear dependency between the hidden layer and the output layer, we have the following system

$$\begin{bmatrix} g_1(\boldsymbol{\beta}_1^{(1)}\mathbf{x}_1^{\mathrm{T}} + b_1) & \cdots & g_M(\boldsymbol{\beta}_M^{(1)}\mathbf{x}_1^{\mathrm{T}} + b_M) \\ \vdots & \ddots & \vdots \\ g_1(\boldsymbol{\beta}_1^{(1)}\mathbf{x}_N^{\mathrm{T}} + b_1) & \cdots & g_M(\boldsymbol{\beta}_M^{(1)}\mathbf{x}_N^{\mathrm{T}} + b_M) \end{bmatrix} \begin{bmatrix} \beta_1^{(2)} \\ \vdots \\ \beta_M^{(2)} \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \tag{3.17}$$

where explicit layer indicator (as a superscript $^{(k)}$) is dropped since we are dealing with only one hidden layer having M neurons. The only parameters in this network are weights $\boldsymbol{\beta}^{(2)}$ which can be obtained by solving the linear system between hidden layer and the output layer, that is, solving the system $\mathbf{H}\boldsymbol{\beta}^{(2)} = \mathbf{y}$. This significantly decreases the training time, while still retaining good generalisation capabilities. It is also shown that this randomisation does not hinder the model from having the universal approximation property [90].

The simple way to find the hidden layer output weights is to use the Moore-Penrose inverse of a matrix, i.e., $\boldsymbol{\beta}^{(2)} = \mathbf{H}^{\dagger}\mathbf{y}$. The original paper where the ELM is proposed uses sine and sigmoid activation function, but the kernels need not be limited to these two only. In several publications in this thesis, the choice has been among the linear, sigmoid and Gaussian kernels. The Gaussian kernels are initialised to some random data points among the samples in data, while inclusion of the linear neurons helps tackle problems that are highly linear. This is particularly important as complexity of the model (number of neurons) can be greatly reduced. In the ELM, the number of data vectors should be higher than the number of neurons in the hidden layer to avoid overfitting. The drawback of the ELM is that the number of neurons must often be much larger than when using backpropagation type algorithms since the input layer weights are

assigned randomly and not learned from the data.

Two scenarios in which the ELM is of great importance are: 1) large amount of data samples, but where the model needs to be trained in reasonable amount of time in order to have a quick insight in the most relevant features and the predictive capabilities based on some validation set; 2) ensemble modelling where a large number of models needs to be trained and later to form a weighted average. The second scenario also arises in variable selection problems where the dimensionality d is high, and in this situation the ELM offers massive improvements in computation time.

Extensions of ELM

After the publication of the seminal paper by Huang et al. [90], there have been many suggestions on how to improve the basic model. These extensions are focused on selecting the appropriate complexity, having more reliable parameter estimation and adapting the ELM network for sequential learning for online types of problems. Among these, the Optimally Pruned ELM (OPELM) [91] provides the most robust results.

OPELM consists of three stages:

1. Construction of the hidden layer weights matrix (Eq. (3.13)) by randomly initialising the weights $\boldsymbol{\beta}^{(1)}$.
2. *Ranking* of the neurons in the hidden layer.
3. Selecting the appropriate number of neurons.

The first stage is the same as in the original ELM. This step provides us with a random mapping to a (usually) higher dimensional space if the number of neurons is larger than the dimensionality of data. This newly transformed data is represented by the $\mathbf{H}$ matrix (Eq. (3.13)), sometimes called the feature mapping or the feature space of the ELM. The second stage consists of ranking these newly created features. For this task, Least Angle Regression (LARS) algorithm is employed, a method used to rank the variables in a linear setting. An important feature of LARS is that the ranking is exact when the problem is linear, that is, the ordering of the variables based on their relevance is correctly identified by the algorithm. The last stage validates how many neurons are needed for the data at hand. This is accomplished via leave-one-out (LOO) cross-validation method.

Cross-validation [1, 4, 20] is one of the most used strategies for evaluating regression models, and provides an immediate comparison between a wide range of different model classes. With the cross-validation, the data is split into folds, where each fold plays a part as a validation set once the model is trained on the remaining folds. After the validation errors are obtained across all the folds, the average is taken as a measurement of generalisation ability for the model. The model with the smallest average validation error is assumed to be the most appropriate for the given data set.

When each data sample is taken to be a single instance in a separate fold, we have what is known as the leave-one-out cross-validation. In the case of least squares linear regression, the LOO error can be computed with a single fit of the linear model using the PRESS statistic [92], which removes the computational burden of training N separate models. If we denote with $\mathbf{P} = \mathbf{H}(\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}$, then the leave-one-out residuals for all the samples are computed with the PRESS formula

$$\tilde{\mathbf{e}}^{\mathrm{LOO}} = \frac{\mathbf{y} - \mathbf{P}\mathbf{y}}{\mathbf{1} - \mathrm{diag}(\mathbf{P})}\,. \tag{3.18}$$

where $\mathrm{diag}(\mathbf{P})$ denotes the main diagonal of $\mathbf{P}$, $\mathbf{1}$ a vector of ones and division is performed element by element.

Leave-one-out cross-validation is appealing to use as it provides an estimate how the model is going to perform on the future data, that is, it gives an estimate of the noise variance in the data and it is possible to compare models from different modelling paradigms directly. Besides CV approach, there are many other *model selection* criteria developed in statistics, with Akaike's Information Criterion (AIC) [93] and the Bayesian Information Criterion (BIC) [94] (also known as Schwarz's criterion) being the two widely used both in statistics and machine learning communities. Both of these criteria penalise the *training error* with an additional complexity term, or

$$\mathrm{criterion} = \mathrm{trainingerror} + \mathrm{Pen}(N, \kappa) \tag{3.19}$$

where $\mathrm{Pen}(N, \kappa)$ is a function of the number of samples in the data and the effective number of adjustable parameters in the model. For the ELM type networks this corresponds to the length of $\boldsymbol{\beta}^{(2)}$ vector or the number of neurons in the hidden layer of the network, plus the bias parameter if it is included. Other model selection criteria also have the similar form with the penalty term in addition to the training error, such as the Hannan-

Quinn criterion [95], Minimum Description Length [96] and many others [97].

Akaike's Information Criterion

Akaike developed a criterion based on the information loss when trying to use an approximate model instead of a "true" model. This loss is computed with the Kullback-Leibler (KL) divergence [98] between a model and an imaginary true model. Akaike has shown that using maximum likelihood approach is a biased estimate of the relative expected KL divergence, and that this bias is asymptotically equal to the number of parameters κ of the considered model. This gives the famous formula (multiplied with -2)

$$\mathrm{AIC} = -2\log(\mathcal{L}(\hat{\theta}|\mathrm{data})) + 2\kappa \tag{3.20}$$

where $\mathcal{L}(\hat{\theta}|\mathrm{data})$ is the likelihood of the data given the parameters of the model θ, and $\hat{\theta}$ is the maximum likelihood estimate of θ for the model. When least-squares approach is adopted under the assumption of normally distributed errors, the AIC criterion can be easily transformed based on the residual sum of squares (RSS) which is written as

$$\mathrm{AIC} = N\log\left(\frac{\mathrm{RSS}}{N}\right) + 2\kappa \tag{3.21}$$

where $\mathrm{RSS}/N = \hat{\sigma}^2$ is an estimate of the noise variance from Eq. (2.2). It is shown that AIC is asymptotically efficient in the sense of selecting the model which achieves the smallest average squared error over the candidate set which is useful when the underlying mechanism is of infinite dimension [99]. However, the AIC tends to choose overly complex models with too large model orders in the finite sample data sets. This issue is addressed by several authors [100, 101] explaining the overfitting problem in the small sample scenarios. They suggested the second-order bias correction term to give the corrected AIC or AICc computed as

$$\mathrm{AICc} = N\log\left(\frac{\mathrm{RSS}}{N}\right) + 2\kappa + \frac{2\kappa(\kappa+1)}{N-\kappa-1}\,. \tag{3.22}$$

The use of AICc is suggested for situations where not enough data is available for stable estimation of the adjustable parameters. The rule of thumb is that AICc should be used when the ratio $N/\kappa < 40$. On the other hand, since the second correction term is quite small for the large values of N it follows that AICc should be used regardless of the number of available samples.

Optimally Pruned kNN

Optimally Pruned k-Nearest Neighbour (OPKNN) shares a similar approach to the OPELM method. The main difference is in the choice of the activation function of the network in the hidden layer, which is performed using the nearest neighbour approach. The search for neighbours takes place in the data space, i.e., considering original variables X^j. This in turn completely removes random initialisation from the model, as computing the neighbours is a deterministic step. k-NN is a simple approach to both regression and classification, where the assumption is that similar training samples have similar outputs. The notion of similarity between the samples is usually based on some metric space, where the Euclidean is the most commonly used.

The hidden matrix $\mathbf{H}$ is constructed in the following way. Denoting with $\Delta(\mathbf{x}_i, \mathbf{x}_k) = \|\mathbf{x}_i - \mathbf{x}_k\|^2$ the Euclidean distance between the two samples $\mathbf{x}_i$ and $\mathbf{x}_k$, these distances can be ordered from the smallest to the largest by fixing one sample $\mathbf{x}_i$, i.e., $\Delta(\mathbf{x}_i, \mathbf{x}_{k_1}) \leq \Delta(\mathbf{x}_i, \mathbf{x}_{k_2}) \leq \Delta(\mathbf{x}_i, \mathbf{x}_{k_N})$. The l-th nearest neighbour for the sample $\mathbf{x}_i$ is $NN(l, i) = k_l$, that is, it is the index of the l-th element in the sorted array of distances. Finally, denoting with $y_{NN(l,i)}$ the output component of the l-th neighbour, the hidden layer matrix can be computed as

$$\mathbf{H} = \begin{bmatrix} y_{NN(1,1)} & y_{NN(2,1)} & \cdots & y_{NN(M,1)} \\ y_{NN(1,2)} & y_{NN(2,2)} & \cdots & y_{NN(M,2)} \\ \vdots & \vdots & \ddots & \vdots \\ y_{NN(1,N)} & y_{NN(2,N)} & \cdots & y_{NN(M,N)} \end{bmatrix}. \tag{3.23}$$

The Euclidean metric can be in principle substituted with any other measure of similarity between the patterns. OPKNN does not suffer from the choice of the activations units, since all of them are based on the nearest neighbours. The only remaining parameter is the number of neurons in the hidden layer, i.e., the number of nearest neighbours. The upper limit in this scenario is given by the number of available samples and it is $N - 1$. In practice, computing the matrix $\mathbf{H}$ can take considerable amount of time if M is chosen to high. Algorithms to speed up the nearest neighbours calculations have already been discussed in Section 2.4.1.

3.3.3 Generative Topographic Mapping

The idea behind Generative Topographic Mapping (GTM) is that many data sets exhibit correlations between the variables which can be cap-

tured by the *latent* or *hidden* variables. The most well known method in this domain is factor analysis [102] which is a linear projection from a lower dimensional latent space to a higher dimensional data space. GTM is proposed as a statistical counterpart for the Self-Organising Map [4,81, 103] method by means of a constrained mixture of Gaussians induced by a low dimensional latent space. Although both methods have been developed for unsupervised learning, such as quantisation, clustering and visualisation tasks, there have been several extensions to adapt both models for the time series data and temporal information. In the GTM case, the temporal aspect is captured with the Hidden Markov Model [104,105]. For the SOM, different approaches have been proposed based on the concatenation of the input and the output vectors [106], associating individual local AR models within each prototype of the SOM [107, 108], and recurrent processing [109–112]. Both the GTM and the SOM offer localised approximation of the data distribution leading to the natural segmentation of the input patterns. This enables easy identification and clustering of the dynamics residing in the series. Another advantage of these topology-preserving models are the simple growing architectures which provide an automatic selection of the appropriate number of prototypes [113].

The main advantage of the GTM over SOM is modelling of the distribution of the data $p(\mathbf{x})$ via the latent variables and a mapping from the latent to the data space $f : \mathbf{T} \rightarrow \mathbf{X}$ where $\mathbf{T}$ is the data representation in the latent space. This mapping is governed by a set of parameters $\mathbf{W}$ of a model which can be in principle any model. The data from the latent space of dimension d_l is then transformed into an d_l-dimensional non-Euclidean manifold embedded in the data space. On the other hand, the SOM is a somewhat heuristic method which is not based on any probabilistic latent variable model.

GTM models the data $\mathbf{x}$ as a mixture distribution

$$p(\mathbf{x}|\mathbf{W}, \alpha) = \sum_{k=1}^{K} p(\mathbf{t}^k) p(\mathbf{x}|\mathbf{t}^k, \mathbf{W}, \alpha) \tag{3.24}$$

where $\mathbf{x}$ is a sample in the data space, α the precision parameter for the Gaussian distribution and K is the number of *prototypes* positioned in the latent space.

The distribution for a point in the data space is modelled as a radially-symmetric Gaussian centred on $f(\mathbf{t}, \mathbf{W})$ with variance α^{-1}

$$p(\mathbf{x}|\mathbf{t},\mathbf{W},\alpha) = \left(\frac{\alpha}{2\pi}\right)^{d/2} \exp\left\{-\frac{\alpha}{2}\|f(\mathbf{t},\mathbf{W}) - \mathbf{x}\|^2\right\} . \tag{3.25}$$

The distribution for the data itself $p(\mathbf{x})$ can be obtained by integrating over the latent variables distribution

$$p(\mathbf{x}|\mathbf{W},\alpha) = \int p(\mathbf{x}|\mathbf{t},\mathbf{W},\alpha)p(\mathbf{t})d\mathbf{t} . \tag{3.26}$$

The choice of the prior distribution for the latent space $p(\mathbf{t})$ is chosen to mimic the SOM, and it is a sum of delta functions centred on the nodes of a regular grid:

$$p(\mathbf{t}) = \frac{1}{K}\sum_{i=1}^{K}\delta(\mathbf{t} - \mathbf{t}_i) \tag{3.27}$$

which in turn gives simple representation of the marginal distribution for the data

$$p(\mathbf{x}|\mathbf{W},\alpha) = \frac{1}{K}\sum_{i=1}^{K} p(\mathbf{x}|\mathbf{t}_i,\mathbf{W},\alpha) . \tag{3.28}$$

The parameters $\mathbf{W}$ and α are obtained using the maximum-likelihood approach, which are based on optimising the data log-likelihood.

Since the model is a mixture model, an Expectation-Maximisation (EM) algorithm is used to find the parameter values. The choice of the mapping f is a generalised linear regression of the form $f(\mathbf{t},\mathbf{W}) = \mathbf{W}\boldsymbol{\phi}(\mathbf{t})$ where the elements $\boldsymbol{\phi}(\mathbf{t})$ consist of certain number of basis functions $\boldsymbol{\phi}_j(\mathbf{t})$. With this approach, the M-step in the EM algorithm corresponds to the solution of a set of linear equations. GTM optimises the data log-likelihood

$$\mathcal{L}(\mathbf{W},\alpha) = \sum_{i=1}^{N} \ln\left(\frac{1}{K}\sum_{k}^{K} p(\mathbf{x}_i|\mathbf{t}_k,\mathbf{W},\alpha)\right) \tag{3.29}$$

with respect to the parameters of a generalised linear regression $\mathbf{W}$ and the precision of the noise α. In the E-step, the responsibility of a mixture component k for a data point $\mathbf{x}_n$ is determined as

$$r^{kn} = p(\mathbf{t}^k|\mathbf{x}_n,\mathbf{W},\alpha) = \frac{p(\mathbf{x}_n|\mathbf{t}_k,\mathbf{W},\alpha)p(\mathbf{t}_k)}{\sum_{j=1}^{K} p(\mathbf{x}_n|\mathbf{t}_j,\mathbf{W},\alpha)p(\mathbf{t}_j)} . \tag{3.30}$$

The M-step involves computing the new weights $\mathbf{W}_{\text{new}}$ and a new value for the precision α_{new}. The weights are computed by the solving linear system

$$\Phi^{\mathrm{T}}\mathbf{G}_{\text{old}}\Phi^{\mathrm{T}}\mathbf{W}_{\text{new}} = \Phi^{\mathrm{T}}\mathbf{R}_{\text{old}}\mathbf{X} \tag{3.31}$$

where Φ is a matrix of basis functions evaluated at points $\mathbf{t}_k$, $\mathbf{X}$ the data itself, $\mathbf{R}$ the responsibilities ($K \times N$ matrix with elements r^{nk}), and $\mathbf{G}$ is a diagonal matrix $K \times K$ with accumulated responsibilities $G_{kk} = \sum_{i=1}^{N} r^{ki}$. The new value for the variance is then computed as

$$\frac{1}{\alpha_{\text{new}}} = \frac{1}{Nd} \sum_{i=1}^{N} \sum_{k=1}^{K} r^{ki} \|\mathbf{W}_{\text{new}} \boldsymbol{\phi}(\mathbf{t}_k) - \mathbf{x}_i\|^2 . \tag{3.32}$$

An important note is that the centres of Gaussians cannot move independently but are related via the mapping f. If this mapping is smooth and continuous, the projections of close points t_i and t_j in the latent space will map onto points $f(t_i)$ and $f(t_j)$ which are close in the data space. This way there exist a topographic ordering that maps the latent grid into the embedding that preserves the grid structure in the data space.

3.3.4 Relevance Learning

In practice, the GTM method has not become as popular as the the SOM since the SOM often provides good enough solutions even though it is more heuristic in nature than the GTM. Another downside of the GTM is the inherent representation of the noise, or to put it differently, the data is displayed without regarding the intended goal of the user. This additional information, or auxiliary data needs to be taken into account and the model has to be modified accordingly. The user is free to specify which information in data is relevant for the current situation in the form of labelled data.

The method based on the GTM where this additional information is included is with the *relevance learning* paradigm [114]. This approach has been developed for the classification purposes with the idea of ranking or finding the *relevance profile* for the features in the data [115]. This profile is obtained via the appropriate metric learning build into the GTM model as the extra learning steps after the EM steps.

Relevance learning is a simple technique to adapt the metric of the prototype-based classifiers. The main idea is to replace the Euclidean metric commonly used with a weighted form

$$\Delta_\lambda(\mathbf{x}_1, \mathbf{x}_2) = \sum_{i=1}^{d} \lambda_i (x_1^i - x_2^i)^2 = (\mathbf{x}_1 - \mathbf{x}_2)^{\mathrm{T}} \Lambda (\mathbf{x}_1 - \mathbf{x}_2) \tag{3.33}$$

which corresponds to using a diagonal covariance matrix $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_d)$ or with a completely new Mahalanobis distance matrix Ω

$$\Delta_\Omega(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 - \mathbf{x}_2)^{\mathrm{T}} \Omega^{\mathrm{T}} \Omega (\mathbf{x}_1 - \mathbf{x}_2)\,. \tag{3.34}$$

This scheme allows both global and local representations, that is, a single matrix might describe the complete model (global approach), or each prototype can have its own local representation given by a dedicated matrix Ω_k, $k = 1, \ldots, K$.

Relevance Learning in GTM

A relevance learning approach can be introduced into the GTM itself, by substituting the Euclidean metric in Eq. (3.25) either with a weighted Euclidean metric (Eq. (3.33)) or with a full covariance matrix $\Omega^{\mathrm{T}}\Omega$ taking into account the correlations between the dimensions (Eq. (3.34)).

Since the GTM is an unsupervised learning algorithm, a suitable way of introducing label mapping from the latent variables or prototypes onto the data points in required. Here we assume the problem is classification, and that data points are labelled where the labels l^i are coming from a finite set. GTM gives the probabilistic classification if the prototypes contain labels themselves via the responsibilities r^{ki}. This implies that the labelling can be performed backwards from the samples onto the prototypes with the following formula

$$C(\mathbf{t}_k) = \arg\max_c \left(\sum_{i|l^i=c}^{N} r^{ki} \right) \tag{3.35}$$

that is, the prototype assumes the label of the highest class/label it is responsible for among all data samples. During the optimisation steps of the GTM, both E and M steps remain the same with the difference that the new metric structure is used when computing the responsibilities (Eq. (3.30)). The optimisation procedure for finding the metric weights is done with a simple stochastic gradient descent of the cost function. For the classification problem, the cost function depends on the distances between the sample $\mathbf{x}_i$ and the two prototypes that are closest to $\mathbf{x}_i$ with one having the same label as $\mathbf{x}_i$ ($C(\mathbf{t}_{k_1}) = l^i$) while the other prototype has a different label ($C(\mathbf{t}_{k_1}) \neq l^i$). For further details and motivations for the specific forms of the cost functions see [115].

3.4 Contributions and Results

In this section, we present some of the contributions of this thesis for the Extreme Learning Machines and the Relevance Learning.

3.4.1 Training in Small Sample Data (Publication III)

Accurate estimation of the hyper-parameters of a model requires substantial number of samples. In certain research domains, such as bioinformatics, chemometrics, or biology connected to marine systems, the number of samples is small, ranging from just a few to a couple of tens. Validation of a model structure and parameters becomes a difficult task. OPELM model relies on the leave-one-out estimates of the output and correspondingly uses the LOO error as a criterion of model complexity. The LOO error itself is a nuisance parameter measuring the noise level present in the data, and due to the random nature of learning can be susceptible to overfitting. Low number of samples poses problems for the LOO error resulting in the estimates with a high variance.

LOO is just one of the model selection criteria introduced in the literature. Small sample data has been studied extensively in statistics, and several solutions have been provided using some measure of complexity, such as corrected AIC [101], corrected Kullback-Information Criterion [116], corrected Hannan-Quinn criterion [97] and Mallow's C_p [117] to name a few.

Publication III deals with the application of the OPELM (OPKKN) to the benthic data from the North Sea. The data consists of only 28 samples collected from the year 1978 to 2005. The benthic data is yearly data measuring abundance, biomass and the number of species of the benthic species in a specific region of the North Sea. The data and environmental modelling is described in more detail in Chapter 4. In this section, we present some results relevant for the model selection for the ELM. Table 3.1 shows the difference between two criteria: leave-one-out cross-validation and Akaike's Information Criterion (plus the corrected version), in terms of the predictive capabilities.

Several data sets from the UCI machine learning repository[1] are used, and the setup is done as follows. Each data set is split into a training set with the low number of samples (20 for all data sets), while the rest of the data is used as a test set. The average of 500 runs of this procedure

[1] `http://archive.ics.uci.edu/ml/`

is taken as an estimated risk under the squared error loss (average test MSE). Both the LOO and the AIC exhibit overfitting in all data sets (the selection of high number of neurons), while the AICc tends to choose less complex models resulting in a more reliable predictive performance. This indicates that a careful choice of the model selection criterion is crucial in situations where the number of samples is low. Model selection is also present when choosing the appropriate number of neurons for the neural network with a single hidden layer, that is, it is tied to the model *structure* selection. Corrected Akaike Information Criterion is a good choice for both of these scenarios which can significantly improve upon the results obtained with the original AIC and the widely used leave-one-out selection.

Data set	Criterion			Error range
	LOO	AIC	AICc	
Auto price	3.95	4.80	2.45	e+07
Delta ailerons	2.33	3.10	0.91	e+00
Housing (Boston)	2.12	7.40	1.17	e+02
Machine CPU	0.51	1.12	0.29	e+05
Servo	6.12	7.51	2.28	e+00
Stocks	8.59	315	2.85	e+01
US Crime	6.74	8.49	2.99	e+05

Table 3.1. Estimated risk for the tested model selection criteria.

3.4.2 Extreme Learning Machine as a Combination Model (Publication VI)

One drawback in the basic Extreme Learning Machine method is the choice of the number of hidden neurons. This has been extensively studied and many solutions have been proposed based on the different model selection criteria [118–120], adapting the output of the neurons making it more stable for regression [121], and the ensemble approaches with many ELMs trained [122]. The ensemble or combination approach tries to avoid the problem by constructing a much larger candidate set, i.e., many models with different number of neurons in the hidden layer, and focusing on finding the appropriate model weights. The aim is to assign low or zero weights for the poor models, while better models (in terms of the generalisation ability) should be given larger weights. Familiar examples of the ensemble modelling are mixture of experts [1], Adaboost [123, 124] and

Bayesian model averaging [125].

Many of the proposed solutions that are based on a specific selection criterion where several architectures are considered (number of neurons ranges from 1 to M) can be categorised into two classes: pruning – starting from a large architecture and then removing unneeded neurons; and constructive – building a large structure from a single or several neurons and gradually adding more neurons. Both strategies are iterative where the procedure is stopped once a condition (usually the validation error) does not improve upon the previous iteration. The output of both strategies is the same – a single model that is deemed "the best" for the data at hand. The idea proposed in Publication VI is to consider all tested architectures as the candidate models and then form a weighted average from all these models instead of picking a single model. The proposed method *Extreme Learning Machine-combination* can be regarded as a class of approaches where different combination methods can be paired with different selection criteria. In the publication, the choice is to use the commonly employed leave-one-out criterion and the Jackknife Model Averaging (JMA) [126]. JMA has been recently proposed as a combination method based on the leave-one-out residuals of the candidate models.

The core idea follows a similar approach as the pruning strategy of the OPELM. The models start with a fixed number of neurons M. The first phase is to construct M models where each model $\mathcal{M}_i$, $i = 1, \ldots, M$ contains i neurons in the hidden layer. The difference is that each model $\mathcal{M}_i$ is a submodel of a larger model $\mathcal{M}_j$ if $i < j$. Table 3.2 shows the results of the predictive capability between the standard ELM model and the proposed extended ELM model. It should be noted that the results are shown for the ELM, where *selection* is done based on the LOO errors, that is, the best model is selected based on the leave-one-out cross-validation procedure. Results are computed for several data sets commonly used for testing regression tasks, and the final outcome is that combining models can provide substantial improvements over the selection strategy in most cases.

These improvements do come at an extra computational load, but the extra computation time is only of slight concern in the small sample data, as the combination method based on the JMA is a quadratic optimisation problem depending only on the number of neurons M. It should be emphasised that the proposed methodology Extreme Learning Machine-combination is applicable to a broader class of regression tasks, and is not

data set	ELM*	ELM(JMA)	(%)
Abalone	12.1	9.14	24.77
Bank_8FM	1.085e−3	1.044e−3	3.79
Boston housing	18.0	15.2	15.92
Breast cancer	1.19e+3	1.43e+3	−20.59
Computer activity	35.8	31.1	12.97
Delta ailerons	2.81e−8	2.74e−8	2.57
Servo	0.729	0.614	15.76
Stocks	0.831	0.716	13.85

Table 3.2. Estimated risk (average mean-squared test error) for the selection and combination in the pure ELM model. The last column indicates improvement in percent.

restricted to the time series data only.

3.4.3 Relevance Learning for Time Series (Publication V)

Publication V extends the relevance learning from the GTM towards regression tasks in time series inspection. The advantage of the approach is two-fold: a versatile model providing long-term predictions and a relevance profile for the time lags that contribute most to the temporal dynamics.

As the GTM is an unsupervised method, Eq. (3.35) gives a way to introduce labelling for prototypes in the latent space. In time series inspection, the labels l^i are replaced with the real values of the actual output y_i, and the posterior labelling is done as follows

$$c(\mathbf{t}_k) = \frac{\sum_{i=1}^{N} r^{ki} y_i}{\sum_{i=1} r^{ki}} \,. \tag{3.36}$$

This produces a smooth regression function for the samples $\mathbf{x}_i \rightarrow l(\mathbf{x}_i)$,

$$l(\mathbf{x}_i) = \sum_{k=1}^{K} r^{ki} c(\mathbf{t}_k) \tag{3.37}$$

and enables computation of the error function (squared error loss)

$$\mathrm{MSE}(\boldsymbol{\lambda}) = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \sum_{k=1}^{K} r^{ki} c(\mathbf{t}_k) \right)^2 - \frac{\gamma}{2} \sum_{i=1}^{d} \exp(-\lambda_d^2) \,. \tag{3.38}$$

The GTM steps remain the same as explained in Section 3.3.4, where the new metric computation involves the scaling coefficients for each dimension λ_d (Eq. (3.33)). The regularisation term $\frac{\gamma}{2} \sum_{i=1}^{d} \exp(-\lambda_d^2)$ $(\gamma \geq 0)$ is added to enforce sparsity in the relevance profile. The coefficients

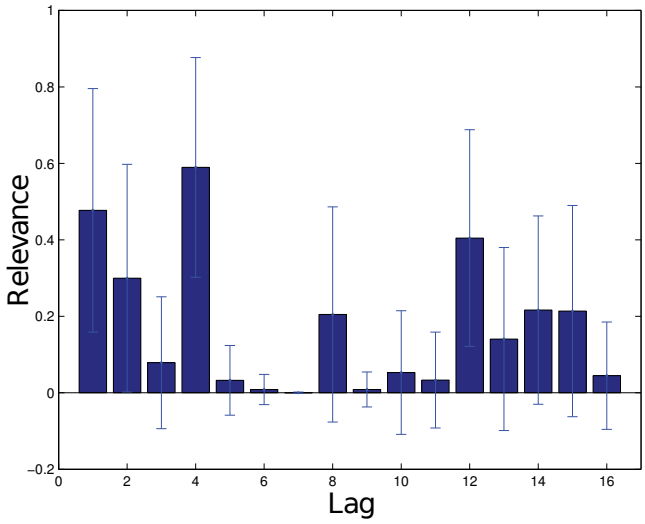

Figure 3.3. Selected time lags for herring time series.

$\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_d]$ can be updated using simple a gradient approach with the normalisation performed afterwards. The whole procedure is summarised as Algorithm 4 which is applied to several time series with the sliding window to produce the data.

Algorithm 4 Relevance learning with GTM for regression.

1: Initialise parameters of GTM
2: **repeat**
3: E-step: determine r^{ki} based on $\|\mathbf{x} - \mathbf{t}\|^2_{\lambda}$ (Eq. (3.30))
4: M-step: determine $\mathbf{W}$ (Eq. (3.31)) and α (Eq. (3.32))
5: label prototypes (Eq. (3.36))
6: adapt λ_d coefficients via gradient descent on MSE(λ)
7: normalise and regularise $\boldsymbol{\lambda}$
8: **until** convergence

One of the time series tested is the Herring data set collected as the seasonal spawning stock biomass of herring from the Baltic Sea main basin measured during 1974-2007. A time window of 16 corresponding to a time span of 4 years is chosen as a regressor size. The proposed method returns several lags as more relevant when doing short-term prediction: previous 3 (Variables 4, 8, 12) seasons we want to predict, while lags 13, 14 and 15 have high value due to possible influx of new adults at the present time, as the period of sexual maturity for herring is around 3 years. The selected lags are displayed in Figure 3.3.

Similar to the approach discussed in Section 3.4.2, the proposed relevance learning method is applicable outside of time series prediction. It can be employed to other regression data sets making the method more general.

4. Application to Marine Systems

This chapter presents one approach to modelling marine ecosystem response with respect to both the global and local climatological factors. The impact of the climate change on local ecosystems has been studied for over two decades, from impact of salinity, inflow of fresh water, nutrients, rainfall and other factors on the local riverine and marine lifeforms.

4.1 Teleconnection Patterns and Climate Indices

Local and regional climates are affected by both the large-scale atmospheric circulation and the surface features [127]. Since the spatial distribution of surface characteristics remains relatively unchanged, it is expected that large-scale climatological changes greatly influence changes in the local climate. Changes in the local climate are often linked to variations in a more global atmospheric circulation. In order to better understand these variations, a suitable *index* or *pattern* can be developed that explains the surface climate changes. The term pattern refers to the *teleconnection pattern* that explains the recurring and persistent, large-scale fluctuations of the pressure and circulation anomalies spanning vast geographical areas. These patterns are usually modes having long time scales, lasting from several days, weeks and sometimes even several consecutive years, thus constituting important part of both the interannual and interdecadal variability of the atmospheric circulation. Some patterns cover the whole continents and oceans and are thus planetary in nature. One of the first developed indices is a "zonal index" proposed by Rossby [128]. Due to the improved understanding of the atmosphere, more useful indices have been established for the use in regional climate. One such example is the Arctic Oscillation index responsible for the northern hemisphere weather circulations.

4.1.1 Arctic Oscillation

The Arctic Oscillation (AO) [129, 130] refers to the opposing pattern of pressure between the Arctic and the northern middle latitudes (around $37 - 45\,^{\circ}N$). If the atmospheric pressure is high in the Arctic, it is generally low in the northern middle latitudes (northern Europe and North America). This case corresponds to the negative phase of the AO. The positive phase happens when the pressures are reversed – low in the Arctic and high in middle latitudes. Having an indicator helps explain certain weather conditions across the northern hemisphere.

In the positive phase, the weather is wetter in Alaska and northern Europe while being drier in western United States and the Mediterranean. During this phase, the weather in eastern US is warmer, but making Greenland colder than normal. In the negative phase, the patterns are reversed. Strong negative phase brings warm conditions to the high latitudes, and colder weather in the middle latitudes.

During the 20th century, the Arctic Oscillation alternated between its positive and negative phases. For a period during the 1970s to mid-1990s, the Arctic Oscillation tended to stay in its positive phase.

4.1.2 North Atlantic Oscillation

The North Atlantic Oscillation (NAO) is one of the most prominent climate patterns in all season in the North Atlantic Ocean. This phenomenon is associated with the fluctuation of differences of the atmospheric pressure at sea level between the Icelandic low and Azores high. Similar to the AO, it goes through the positive and negative phases. Both phases are basin-wide in intensity and location of the North Atlantic jet stream and storm track, resulting in the temperature and precipitation changes covering an area from the eastern North America to the western and central Europe [131].

In the positive phase, above-average temperatures are expected in easter US and across northern Europe. This phase is also associated with the higher precipitation period over northern Europe in winter, and below-average precipitation over southern and central Europe. The opposite patterns of temperature and precipitation are experienced in the case of the negative phase. During the prolonged periods of one dominating phase, the anomalies can be carried over into the central Russia and north-central Siberia.

An interesting feature of the NAO is its considerable interseasonal and interannual variability, with possibly prolonged periods of either positive or negative phase. For example, the negative phase was prominent between mid-1950's through 1978/1979 in which the positive phase (as a seasonal mean) was almost absent from the observations. After 1978/1979 years, an abrupt transition toward the dominating positive phase has occurred which remained until mid 1990's. During this period, a substantial negative phase appeared only twice. Starting from the end of 1995, the NAO has returned toward predominantly negative phase.

4.1.3 Other Indices

Besides the two mentioned indices, there are other indices [132] explaining weather phenomena for the southern hemisphere, out of which the most dominant is the Southern Oscillation which gives an indication of the development and intensity of the El Niño and La Niña events in the Pacific Ocean. It is calculated using the pressure differences between Tahiti and Darwin. Some other indices include: Pacific North American Index (PNA), Pacific/North Pacific Oscillation (EP/NP), North Pacific pattern (NP), Pacific Decadal Oscillation (PDO), Antarctic Oscillation (AAO), Atlantic multidecadal Oscillation (AMO). A comprehensive list is kept and updated by the Earth System Research Laboratory[1].

4.2 Marine Ecosystems

The impact of the climate patterns on regional conditions has been studied extensively. These include influence of rainfall [133, 134], ice-sheet cover [135, 136], fresh water inflow [137, 138] and salinity [139] to name a few. On the other hand, most of the faunal and floral biodiversity is strongly dependent on the regional levels salinity, nutrients, oxygen, nitrogen among many others. For example, in the Baltic Sea, species composition of zooplankton generally follow changes in the salinity levels [140]. Salinity, in turn is directly affected by the climatological factors via river run-off and the influx of saline water from the North Sea [141].

A lot of research has been devoted to understanding the connections between the climate variability and various levels of marine ecosystem. Several publications have shown that major parts of the biological variability

[1] http://www.esrl.noaa.gov/psd/data/climateindices/list/

can be attributed to the physical fluctuations. Certain species show linear response to the climate variability in the northern hemisphere, such as zooplankton in the Northeast Atlantic and the Baltic Sea [142] and different species of fish [143]. Most notable is the use of NAO in the Atlantic sector for predicting terrestrial [144], freshwater [145] and marine ecosystems [146]. Previous results indicated that atmospheric winter circulation is a reasonable indicator of variabilities in macrofauna for the following spring in the southern North Sea [147]. The suspected mediator between the two is the sea surface temperature (SST) which is highly correlated to the NAO index [148].

The main interest is relating the global climate pattern, such as the AO or the NAO, to the time series of interest. This approach has been applied to many marine systems, with the basic idea of correlating potential climate variables with the regional observations [147]. The basic procedure is outlined below:

1. Let $\mathbf{y}$ be marine time series, such as fish, zooplankton or benthos.
2. Let $\mathbf{z}_j$ indicate potential climate factors.
3. For all combinations of $\mathbf{z}_j$ and $\mathbf{y}$:
   (a) Fit a model
   (b) Validate
4. Two tests are done:
   (a) Validation accuracy should be high enough.
   (b) Ecological plausibility between two paired time series.
5. If both tests are satisfied, this indicates potential relationship between the two tested series $\mathbf{z}_j$ and $\mathbf{y}$.

The standard practice in oceanography and biology is done with a single factor $\mathbf{z}_j$ since the goal is identification of the *major* driving force behind $\mathbf{y}$. This is Step 3. in the previously described procedure. However, certain combinations are excluded from the consideration since the major climatological factors, such as the AO and the NAO, are closely related and usually only one is taken into account. In Section 4.3, a novel method based on the multiple factors is proposed that improves upon the methodology based on a single factor.

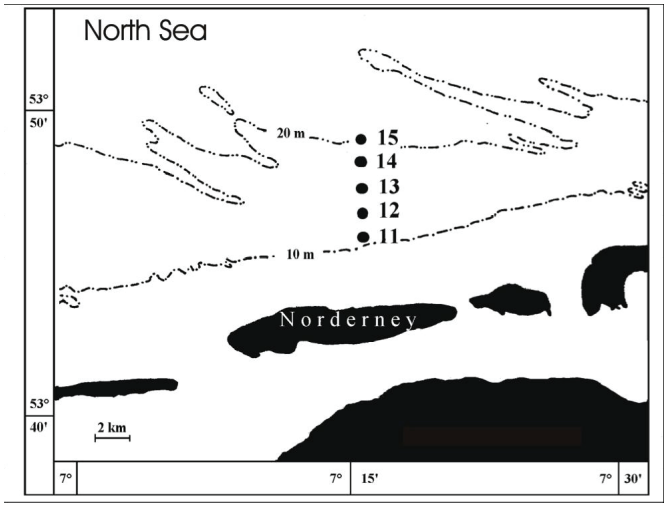

Figure 4.1. Study area comprising of five stations of the island of Norderney.

4.3 Multifactor Approach (Publication III)

As mentioned, the NAO was thought to be a reliable indicator of variabilities in macrofauna in the southern North Sea. This relationship breaks after year 2000 due to the climate shift into a different mode of operation [149]. In this situation, it is no longer possible to predict the biological variability with a linear model. To solve the issue, two possibilities exist: creating a new index or employing a different model for the task. Publication III proposes a modified OPELM and OPKNN model for this problem. Another important issue is that a single global pattern is often not sufficient to explain changes in the local ecosystem. For this reason, several other measurements are used to improve the forecast.

4.3.1 Data

Benthic macrofaunal samples are collected in the 2nd quarter during the period 1978-2005 in the sublitoral zone of the island of Norderney in the North Sea at five different stations located in water depths between 12 meters and 20 meters. The location of these stations is indicated in Figure 4.1. Each measurement is taken to be yearly representation of the state of the benthic species in the region. This gives a total of 28 measurements for the period in question.

The standardized procedure is used to collect the samples from the sea: a 0.2m^2 van Veen grab is used for sampling. A single grab is taken at each of the five stations. After that, the samples are sieved over 0.63mm

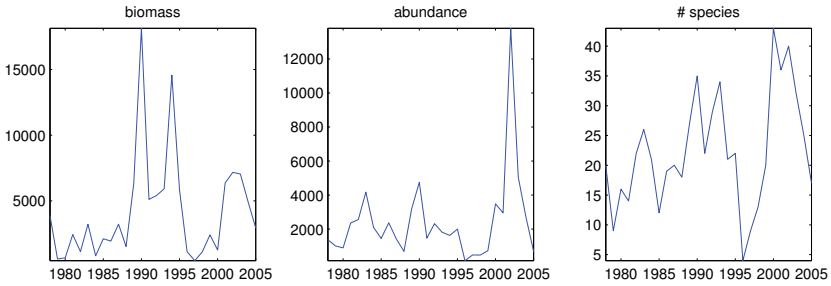

Figure 4.2. Benthic time series (abundance, biomass and species number) in the southern North Sea.

mesh size and fixed in 4% buffered in formaldehyde. After sorting, the organisms are preserved in 70% alcohol. Biomass is determined as ash-free dry weight per m^2. Samples are dried for 24h at 85°Cand burned for 6h at 485°C. The species number, abundance and biomass from the five stations are pooled and treated as replicates for the area. All three time series are displayed in Figure 4.2. These three series are the focus of the Publication III.

In addition to the global climate factors, several local or regional time series are used to improve the forecast. As shown in [147], there is a phase lag between the climate variability during the winter time and the response in macrozoobenthos in the North Sea in spring. For this reason, all the measurements are taken to be the *winter averages* over the period of four months – December, January, February and March (DJFM). The response during the summer and autumn is more unreliable due to the increased predator-prey interactions making the system more chaotic. The tested time series are summarised in Table 4.1.

The data is taken from two major bodies for scientific research. The first one is the joint project between the National Centers for Environmental Prediction (NCEP) and the National Center for Atmospheric Research[2] (NCAR) which continually updates gridded data set representing the state of the Earth's atmosphere covering the whole globe. The "Reanalysis" data sets [150] are created by assimilating the climate observations using the same climate model throughout the entire reanalysis period (1948-present). The observations are coming from many different sources, such as ships, planes, satellites, ground stations, radar and many more and the data is updated four times a day.

The second source of data comes from the International Council for the

[2] `http://www.esrl.noaa.gov/psd/data/reanalysis/reanalysis.shtml`

Climate indices:
• Arctic Oscillation (AO)
• North Atlantic Oscillation (NAO)
• Atlantic Multidecadal Oscillation (AMO)
Regional observations:
• area averaged monthly meridional wind anomalies (1948–2010) in the southern North Sea (53°-56° N,2°-9° E) from NCEP/NCAR reanalysis
• monthly precipitation rate anomalies (1948–2010) averaged over the area 50°-57° N, 4° W-9° E from NCEP/NCAR reanalysis
• area averaged monthly sea surface temperatures anomalies (1948–2009) in the southern North Sea (53-56° N, 2° W-9° E) from NCEP/NCAR reanalysis
• salinity from ICES (Marsden square 96668)
• temperature from ICES (Marsden square 96668)
• weekly sea surface temperature data for the German Bight from 1968 to 2007 south of 55.5° N and east of 6.5° E

Table 4.1. Time series tested against benthic macrofauna species.

Exploration of the Sea[3] (ICES), an intergovernmental organization with the objective of advancing scientific knowledge of the marine environment and its living resources. The main goal behind ICES considers how human activities affect the marine ecosystem and vice versa. ICES manages a number of large data set collections related to the marine environment which span areas of the North East Atlantic, the Baltic Sea, the Greenland Sea and the Norwegian Sea. These data collections cover several research domains: biological communities, contaminants and biological effects, plankton data, ocean physics and ocean chemistry, and fish predation.

4.3.2 Method

The interdecadal variability of climate patterns is an important aspect of atmospheric circulation, and for this reason a long time lag is used when forming the regressor. Each time series is composed as the winter averages over the available values from the databases and 12 values are used as the input to the model. These include the winter prior to the

[3] `www.ices.dk`

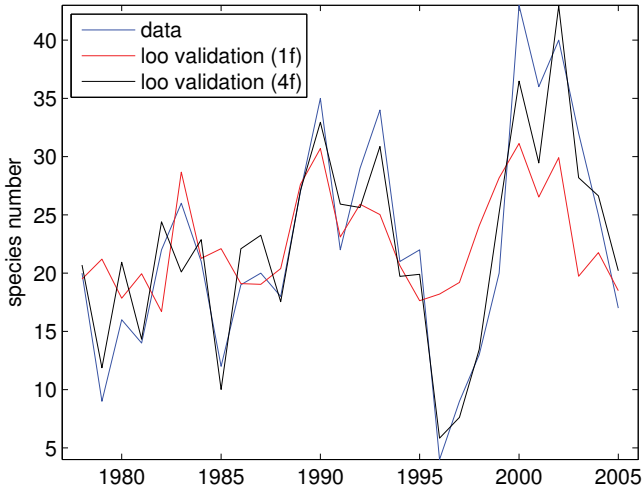

Figure 4.3. Leave-one-out validation comparison between one factor approach (1f) and four factor approach (4f). The latter method with additional information (temperature, wind, precipitation) offers improved accuracy across the entire period.

spring measurements of the benthos, plus 11 years before.

An important aspect of relating climate factors towards any ecological measurements is the *exogenous* model, where the target variable is solely regressed on the other inputs, i.e., $y_t = f(z_t^1, z_{t-1}^1, \ldots, z_t^2, z_{t-1}^2, \ldots)$ where $\mathbf{z}^1$ and $\mathbf{z}^2$ are two exogenous time series. Although this might seem a difficult task, it is useful in specific cases where the number of samples is quite low (which is case we are examining here). If a regressor is build using Eq. (3.4) for each time lag introduced one less sample is available for training the model. In certain cases, the periodicity of the series can be large which would require several time lags to be used. This in turn reduces the number of samples making the model fitting procedure very difficult. Since many time series are at our disposal, the final data set contains several tens of variables. For this reason, a wrapper approach (Section 2.1) has been used to select the most important time lags. As the number of possible solutions is quite large, the Forward-Backward Search is used to find the local minimum, where the criterion used is the leave-one-out validation error. Two models are used for this task – OPELM and OPKNN. Since OPELM is inherently stochastic in nature, the initialisation for this model is done a 100 times. Thus for each selection of variables, 100 models are trained and the average of the validation errors is used as the final performance for that particular feature set. OPKNN

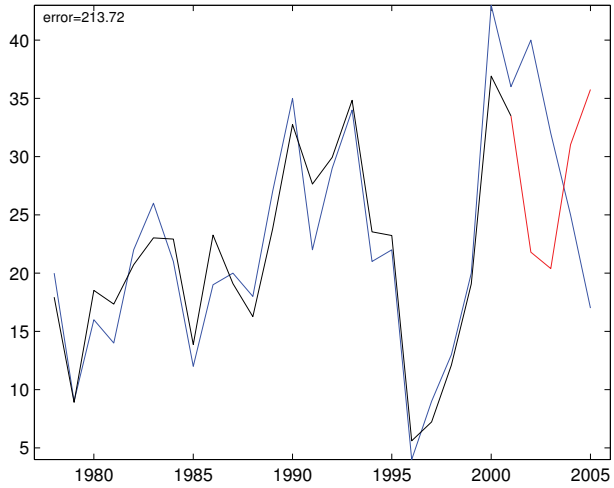

Figure 4.4. Species number times series (blue), LOO validation curve (black) and prediction for the period 2000-2005 (red)

.

is deterministic and only needs to be run once. The whole procedure is repeated 1000 times to have somewhat stable set of relevant features (time lags). A single model (either OPELM or OPKNN) uses different criterion for the complexity (number of neurons) which is the corrected Akaike's Information Criterion (Section 3.3.2).

Figure 4.3 shows the comparison between using only a single factor (climate index – Arctic Oscillation) and four factors (Arctic Oscillation, meridional wind, precipitation over the region and temperature from the ICES database) with respect to the species number time series. Other time series indicated in Table 4.1 are not used since their influence did not bring any improvement over the four mentioned. What is shown are the leave-one-out predictions over the entire period of 28 years. As depicted, using additional information from the local observation can greatly improve the trained model. The graphs are for the OPKNN since the results for the OPELM are more variable due to the random nature of the first layer weights. Leave-one-out predictions are based on the chosen model which uses the leave-one-out residuals to compute the validation error as a measure of fit. The data is formatted according to the Eq. (3.4), that is, a regressor matrix is formed based on the time series and the chosen lag h. The temporality is embedded into this matrix, and any regression method can be applied to obtain the LOO predictions. OPKNN and

OPELM models are suitable since the LOO residuals can be easily computed as explained in Section 3.3.2.

Finally, Figure 4.4 shows the results when the actual model is used for short-term prediction. This is a one step-ahead setup for the period of 2000-2006 with a *single* model trained on the features selected from the procedure described previously. Unfortunately, the model fails to provide a reliable prediction for the next 6 years even in one-step ahead scenario and quite accurate validation predictions. This can be attributed to the heavy climate shifts, making the system inherently unstable and difficult to predict with quite small number of samples.

5. Conclusion

In this dissertation, two problems in machine learning domain are tackled. One is that of variable selection, while the other falls broadly into the time series prediction category.

Simply stated, variable selection is a procedure that tries to identify *relevant* features or variables that contribute the most to the target or output variable. The whole procedure can be viewed from different perspectives: pure selection, scaling and projection. Comparing to pure inclusion/exclusion selection approach, in the scaling variant each variable is weighted according to the importance level. This generalises the problem, but at the same time increases the complexity by introducing the new solution space. Going even further is the projection method, enabling the user to completely transform or project the data into a new feature space. All three perspectives are tackled in the thesis with the Delta test as the main and only search criterion. Several solutions for variable selection are proposed that are based on Genetic Algorithms and Tabu Search optimisation methods. One of interesting points is the combined scaling and projection approach which extends or *increases* the dimensionality of the data. This may sound counter intuitive as the goal is reduction of number of variables. On the other hand, if only a few variable are kept from the scaling part, the additional feature(s) (projection part) that are linear combinations do bring valuable information improving the learning stage. One might think that having a projection requires specifying the number of dimensions for the projection matrix which constitutes another parameter to be optimised. However, a simple procedure enables the automatic selection of the projection dimension which removes the burden of either cross-validation or the intervention from the user. Several methods are proposed that enable faster execution of the variable selection with the Delta test and these include: 1) parallel implementations on both clus-

ter of homogeneous and heterogeneous clusters of computers; 2) use of approximate nearest neighbours when computing the distances between the input samples and 3) multistart strategy in large sample data since the optimisation landscape contains only small number of local minima. Each of the proposed solutions try to cope with the situations where the number of variables is high enough to prohibit any kind of exhaustive search.

The second problem of time series prediction is of great practical importance. It has been studied substantively and the usual approach is to form a regressor matrix which is subsequently used by a training model. In the thesis, three methods are proposed that can be used for the purpose of prediction. Two are based on the Extreme Learning Machine neural network model. The first suggestion is the choice of a model selection criterion for the complexity of the network when the number of samples is extremely small. This situation arises in many areas of biology and poses great difficulty for majority of machine learning techniques. The suggested corrected Akaike's Information Criterion is specifically designed for these scenarios and it greatly improves the stability of both the OPELM and OPKKN. The second improvement comes in the form of model averaging directly in the ELM model itself. Most of the pruning or additive solutions for the level of complexity of the network choose only a single "best" model. The solution proposed in the thesis is the take into consideration *all* the models and do the model weighting from all the constructed substructures. This is accomplished with the use of Jackknife Model Averaging method and leave-one-out cross-validation errors which provides better results over the selection approach alone. For future research possibilities, different weighting combination techniques and selection criteria can be investigated and compared to the current setup. Finally, the extension of the Generative Topographic Mapping to include relevance learning is introduced for time series inspection. Relevance learning replaces the Euclidean metric in the GTM with a weighted distance computation. This combination allows time lags inspection (variable scaling) and long-term prediction at the same time.

The last contribution on the thesis touches upon environmental modelling. Specific interest is devoted to modelling benthic data with respect to both global climate factors and regional observations. The suggested approach is to combine both types of information to form a multifactor index that provides improved function estimation over the usual

approach with one climate index. This multi-factor approach is attracting researchers in the environmental community [151]. Future research possibilities include relating marine ecosystem time series with other potential predictors, such as nutrients or fresh-water inflow. Another difficult task worth investigating is the long-term prediction for these small sample time series.

Bibliography

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] E. Alpaydin, *Introduction to Machine Learning*. MIT Press, 2nd ed., 2010.

[3] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Adaptive Computation and Machine Learning, MIT Press, 2012.

[4] S. S. Haykin, *Neural Networks and Learning Machines*. Pearson Education, 3rd ed., 2009.

[5] Y. Bengio, "Learning deep architectures for ai," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[6] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[7] R. Salakhutdinov and G. E. Hinton, "Deep boltzmann machines," in *International Conference on Artificial Intelligence and Statistics (AISTATS 2009)*, vol. 5 of *JMLR Proceesings*, pp. 448–455, 2009.

[8] G. Gan, C. Ma, and J. Wu, *Data Clustering: Theory, Algorithms, and Applications*. Siam, 2007.

[9] I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, "Feature extraction," *Foundations and applications*, 2006.

[10] J. A. Lee and M. Verleysen, *Nonlinear Dimensionality Reduction*. Springer, 2007.

[11] F. Ricci, L. Rokach, and B. Shapira, *Introduction to Recommender Systems Handbook*. Springer, 2011.

[12] A. C. Rencher and W. F. Christensen, *Methods of Multivariate Analysis*. Wiley, 3rd ed., 2012.

[13] E. Eirola, E. Liitiäinen, A. Lendasse, F. Corona, and M. Verleysen, "Using the delta test for variable selection," in *European Symposium on Artificial Neural Networks (ESANN 2008)*, pp. 25–30, 2008.

[14] E. Liitiäinen, F. Corona, and A. Lendasse, "Nearest neighbor distributions and noise variance estimation," in *European Symposium on Artificial Neural Networks (ESANN 2007)*, pp. 67–72, 2007.

[15] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers & Operations Research*, vol. 13, no. 5, pp. 533–549, 1986.

[16] F. Glover, "Tabu search part i," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

[17] M. D. Vose, *The Simple Genetic Algorithm: Foundations and Theory*. MIT Press, 1999.

[18] M. Mitchell, *An Introduction to Genetic Algorithms*. MIT press, 1998.

[19] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[20] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer Series in Statistics, Springer, 2001.

[21] M. Verleysen and D. François, *The curse of dimensionality in data mining and time series prediction*, vol. 3512 of *Lecture Notes in Computer Science*, pp. 758–770. Springer, 2005.

[22] I. Jolliffe, *Principal Component Analysis*. Wiley Online Library, 2005.

[23] E. I. George, "The variable selection problem," *Journal of the American Statistical Association*, vol. 95, no. 452, pp. 1304–1308, 2000.

[24] A. E. Raftery, "Bayesian model selection in social research," *Sociological methodology*, vol. 25, pp. 111–164, 1995.

[25] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, pp. 273–324, 1997.

[26] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *The Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[27] H. Liu, H. Motoda, and L. Yu, "A selective sampling approach to active feature selection," *Artificial Intelligence*, vol. 159, no. 1–2, pp. 49–74, 2004.

[28] L. Yu and H. Liu, "Feature selection for high-dimensional data: a fast correlation-based filter solution," in *International Conference on Machine Learning (ICML 2003)*, pp. 856–863, 2003.

[29] M. A. Hall, "Correlation-based feature selection for discrete and numeric class machine learning," in *International Conference on Machine Learning (ICML 2000)*, pp. 359–366, Morgan Kaufmann Publishers, 2000.

[30] B. Frénay, G. Doquire, and M. Verleysen, "Is mutual information adequate for feature selection in regression?," *Neural Networks*, vol. 48, pp. 1–7, 2013.

[31] M. Verleysen, F. Rossi, and D. François, "Advances in feature selection with mutual information," in *Similarity-Based Clustering*, vol. 5400 of *Lecture Notes in Computer Science*, pp. 52–69, Springer, 2009.

[32] R. Battiti, "Using mutual information for selecting features in supervised neural network learning," *IEEE Transactions on Neural Networks*, vol. 5, pp. 537–550, July 1994.

[33] S. Arlot and P. Massart, "Data-driven calibration of penalties for least-squares regression," *Journal of Machine Learning Research*, vol. 10, pp. 245–279, 2009.

[34] Y. Kim, W. N. Street, and F. Menczer, "Feature selection in unsupervised learning via evolutionary search," in *International Conference on Knowledge Discovery and Data Mining (KDD 2000)*, pp. 365–369, 2000.

[35] J. Yang and V. Honavar, "Feature subset selection using a genetic algorithm," in *Feature Extraction, Construction and Selection*, vol. 453 of *Springer International Series in Engineering and Computer Science*, pp. 117–136, 1998.

[36] I.-S. Oh, J.-S. Lee, and B.-R. Moon, "Hybrid genetic algorithms for feature selection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1424–1437, 2004.

[37] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan, "An introduction to mcmc for machine learning," *Machine Learning*, vol. 50, no. 1–2, pp. 5–43, 2003.

[38] H. Zhang and G. Sun, "Feature selection using tabu search method," *Pattern Recognition*, vol. 35, no. 3, pp. 701–711, 2002.

[39] T. A. Feo and M. G. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, vol. 6, no. 2, pp. 109–133, 1995.

[40] M. G. Resende, "Greedy randomized adaptive search procedures," in *Encyclopedia of Optimization* (C. A. Floudas and P. M. Pardalos, eds.), pp. 913–922, Springer, 2001.

[41] K. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Springer, 2006.

[42] A. A. Freitas, "A Survey of Evolutionary Algorithms for Data Mining and Knowledge Discovery," in *Advances in Evolutionary Computing*, pp. 819–845, Springer, 2003.

[43] M. Dorigo and M. Birattari, "Ant Colony optimization," in *Encyclopedia of Machine Learning*, pp. 36–39, Springer, 2010.

[44] T. Hastie *et al.*, "Forward stagewise regression and the monotone lasso," *Electronic Journal of Statistics*, vol. 1, pp. 1–29, 2007.

[45] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, pp. 267–288, 1996.

[46] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *The Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[47] F. Glover, "Tabu search part ii," *ORSA Journal on Computing*, vol. 2, pp. 4–32, 1990.

[48] F. Xhafa, J. Carretero, B. Dorronsoro, and E. Alba, "A tabu search algorithm for scheduling independent jobs in computational grids," *Computing and Informatics*, vol. 28, no. 2009, pp. 1001–1014, 2009.

[49] J. Brandao, "A tabu search algorithm for the open vehicle routing problem," *European Journal of Operational Research*, vol. 157, no. 3, pp. 552–564, 2004.

[50] S. Scheuerer, "A tabu search heuristic for the truck and trailer routing problem," *Computers & Operations Research*, vol. 33, no. 4, pp. 894–909, 2006.

[51] F. Glover, "Parametric tabu-search for mixed integer programs," *Computers & Operations Research*, vol. 33, no. 9, pp. 2449–2494, 2006.

[52] K. S. Al-Sultan and M. A. Al-Fawzan, "A tabu search hooke and jeeves algorithm for unconstrained optimization," *European Journal of Operational Research*, vol. 103, no. 1, pp. 198–208, 1997.

[53] S. Kirkpatrick, "Optimization by simulated annealing: quantitative studies," *Journal of Statistical Physics*, vol. 34, no. 5–6, pp. 975–986, 1984.

[54] F. Glover and F. Laguna, *Tabu Search*. Kluwer Academic Publishers, 1997.

[55] A. P. Engelbrecht, *Computational Intelligence: An Introduction*. John Wiley & Sons, 2nd ed., 2002.

[56] D. Goldberg and K. Sastry, *Genetic Algorithms: The Design of Innovation*. Springer, 2007.

[57] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, vol. 4, no. 2, pp. 65–85, 1994.

[58] R. W. Morrison and K. A. De Jong, "Measurement of population diversity," in *International Conference on Artificial Evolution*, vol. 2310 of *Lecture Notes in Computer Science*, (Le Creusot, France), pp. 31–41, 2001.

[59] J. J. Grefenstette, "Parallel adaptive algorithms for function optimization," Technical Report TCGA CS-81-19, Department of Engineering Mechanics, University of Alabama, Vanderbilt University, 1981.

[60] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, 2001.

[61] K. Miettinen, *Nonlinear Multiobjective Optimization*, vol. 12 of *International Series in Operations Research and Management Science*. Kluwer Academic Publishers, 1999.

[62] C.-K. Goh and K. C. Tan, *Evolutionary Multi-Objective Optimization in Uncertain Environments: Issues and Algorithms*, vol. 186 of *Studies in Computational Intelligence*. Springer, 2009.

[63] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[64] K.-L. Du and M. N. S. Swamy, *Neural Networks and Statistical Learning*. Springer, 2013.

[65] D. Evans and A. J. Jones, "A proof of the gamma test," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 458, no. 2027, pp. 2759–2799, 2002.

[66] S. Berchtold, D. A. Keim, and H.-P. Kriegel, "The x-tree: an index structure for high-dimensional data," in *International Conference on Very Large Data Bases (VLDB 1996)*, pp. 28–39, 1996.

[67] P. B. Callahan and S. R. Kosaraju, "A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields," *Journal of the ACM*, vol. 42, no. 1, pp. 67–90, 1995.

[68] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[69] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[70] D. Draper, "Assessment and propagation of model uncertainty (with discussion)," *Journal of the Royal Statistical Society: Series B*, vol. 57, no. 1, pp. 45–97, 1995.

[71] E. L. R. Fernandes and C. C. Ribeiro, "Using an adaptive memory strategy to improve a multistart heuristic for sequencing by hybridization," in *International Workshop on Experimental and Efficient Algorithms (WEA 2005)*, vol. 3503 of *Lecture Notes in Computer Science*, pp. 4–15, 2005.

[72] T. Jansen, "On the analysis of dynamic restart strategies for evolutionary algorithms," in *International Conference on Parallel Problem Solving from Nature (PPSN 2002)*, vol. 2439 of *Lecture Notes in Computer Science*, pp. 33–43, 2002.

[73] T. L. James, C. Rego, and F. Glover, "Multistart tabu search and diversification strategies for the quadratic assignment problem," *IEEE Transactions on Systems, Man and Cybernetics (Part A)*, vol. 39, no. 3, pp. 579–596, 2009.

[74] V. Garcia, E. Debreuve, and M. Barlaud, "Fast k nearest neighbor search using gpu," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW 2008)*, pp. 1–6, 2008.

[75] G. E. Box, G. M. Jenkins, and G. C. Reinsel, *Time series analysis: forecasting and control*. John Wiley & Sons, 2013.

[76] P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting*. Springer Texts in Statistics, Taylor & Francis, 2nd ed., 2002.

[77] A. Sorjamaa, J. Hao, N. Reyhani, Y. Ji, and A. Lendasse, "Methodology for long-term prediction of time series," *Neurocomputing*, vol. 70, no. 16–18, pp. 2861–2869, 2007.

[78] A. Sorjamaa and A. Lendasse, "Time series prediction as a problem of missing values: application to estsp2007 and nn3 competition benchmarks," in *International Joint Conference on Neural Networks (IJCNN 2007)*, pp. 2948–2953, 2007.

[79] E. Eirola and A. Lendasse, "Gaussian mixture models for time series modelling, forecasting, and interpolation," in *International Symposium on Intelligent Data Analysis (IDA 2013)*, vol. 8207 of *Lecture Notes in Computer Science*, pp. 162–173, 2013.

[80] C. Chatfield, *Time-Series Forecasting*. CRC Press, 2000.

[81] F. M. Ham and I. Kostanic, *Principles of Neurocomputing for Science and Engineering*. McGraw Hilll, 2001.

[82] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[83] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psycological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[84] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[85] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[86] G. Hinton, D. Rumelhart, and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[87] J. Moody, S. Hanson, A. Krogh, and J. A. Hertz, "A simple weight decay can improve generalization," *Advances in Neural Information Processing Systems*, vol. 4, pp. 950–957, 1995.

[88] L. Prechelt, "Automatic early stopping using cross validation: quantifying the criteria," *Neural Networks*, vol. 11, no. 4, pp. 761–767, 1998.

[89] G. bin Huang, Q. yu Zhu, and C. kheong Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *International Joint Conference on Neural Networks (IJCNN 2004)*, pp. 985–990, IEEE, 2004.

[90] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.

[91] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, "Op-elm: optimally-pruned extreme learning machine," *IEEE Transactions on Neural Networks*, vol. 21, no. 1, pp. 158–162, 2010.

[92] D. M. Allen, "The relationship between variable selection and data augmentation and a method for prediction," *Technometrics*, vol. 16, no. 1, pp. 125–127, 1974.

[93] H. Akaike, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 716–723, 1974.

[94] G. Schwarz, "Estimating the dimension of a model," *The Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[95] E. Hannan and B. Quinn, "The determination of the order of an autoregression," *Journal of the Royal Statistical Society: Series B*, pp. 190–195.

[96] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[97] A. D. R. McQuarrie and C.-L. Tsai, *Regression and Time Series Model Selection*. World Scientific, 1998.

[98] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[99] J. Shao, "An asymptotic theory for linear model selection," *Statistica Sinica*, vol. 7, pp. 221–264, 1997.

[100] N. Sugiura, "Further analysis of the data by Akaike's information criterion and the finite corrections," *Communications in Statistics – Theory and Methods*, vol. 7, no. 1, pp. 13–26, 1978.

[101] C. M. Hurvich and C.-L. Tsai, "Bias of the corrected aic criterion for underfitted regression and time series models," *Biometrika*, vol. 78, no. 3, pp. 499–509, 1991.

[102] A. L. Comrey and H. B. Lee, *A First Course in Factor Analysis*. Psychology Press, 2013.

[103] T. Kohonen, *Self-Organising Maps*. Springer, 2001.

[104] C. Bishop, G. Hinton, and I. Strachan, "Gtm through time," in *International Conference on Artificial Neural Networks*, pp. 111–116, IEE, 1997.

[105] I. Olier and A. Vellido, "A variational formulation for gtm through time," in *International Joint Conference on Neural Networks (IJCNN 2008)*, pp. 516–521, IEEE, 2008.

[106] G. A. Barreto and A. F. R. Araújo, "Identification and control of dynamical systems using the self-organizing map," *IEEE Transactions on Neural Networks*, vol. 15, no. 5, pp. 1244–1259, 2004.

[107] J. Vesanto, "Using the som and local models in time-series prediction," in *Workshop on Self-Organizing Maps (WSOM 1997)*, pp. 209–214, 1997.

[108] G. A. Barreto, J. C. Mota, L. G. Souza, and R. A. Frota, "Nonstationary time series prediction using local models based on competitive neural networks," in *Innovations in Applied Artificial Intelligence* (B. Orchard, C. Yang, and M. Ali, eds.), vol. 3029 of *Lecture Notes in Computer Science*, pp. 1146–1155, Springer Berlin Heidelberg, 2004.

[109] G. Chappell and J. Taylor, "The temporal kohonen map," *Neural Networks*, vol. 6, pp. 441–445, 1993.

[110] M. Hagenbuchner, A. Sperduti, and A. Tsoi, "A self-organizing map for adaptive processing of structured data," *IEEE Transactions on Neural Networks*, vol. 14, pp. 191–505, 2003.

[111] M. Varsta, J. Heikkonen, J. Lampinen, , and J. Milán, "Temporal kohonen map and recurrent self-organizing map: analytical and experimental comparison," *Neural Processing Letters*, vol. 13, no. 3, pp. 237–251, 2001.

[112] M. Strickert and B. Hammer, "Merge som for temporal data," *Neurocomputing*, vol. 64, pp. 39–71, 2005.

[113] G. A. Barreto, "Time series prediction with the self-organizing map: A review," in *Perspectives of Neural-Symbolic Integration* (B. Hammer and P. Hitzler, eds.), vol. 77 of *Studies in Computational Intelligence*, pp. 135–158, Springer Berlin Heidelberg, 2007.

[114] B. Hammer and T. Villmann, "Generalized relevance learning vector quantization," *Neural Networks*, vol. 15, no. 8, pp. 1059–1068, 2002.

[115] A. Gisbrecht and B. Hammer, "Relevance learning in generative topographic mapping," *Neurocomputing*, vol. 74, no. 9, pp. 1351–1358, 2011.

[116] A.-K. Seghouane and M. Bekara, "A small sample model selection criterion based on kullback's symmetric divergence," *IEEE Transactions on Signal Processing*, vol. 52, no. 12, pp. 3314–3323, 2004.

[117] C. L. Mallows, "Some comments on cp," *Technometrics*, vol. 15, no. 4, pp. 661–675, 1973.

[118] Y. Lan, Y. C. Soh, and G.-B. Huang, "Two-stage extreme learning machine for regression," *Neurocomputing*, vol. 73, no. 16–18, pp. 3028–3038, 2010.

[119] Y. Lan, Y. C. Soh, and G.-B. Huang, "Constructive hidden nodes selection of extreme learning machine for regression," *Neurocomputing*, vol. 73, no. 16–18, pp. 3191–3199, 2010.

[120] Y. Miche and A. Lendasse, "A faster model selection criterion for op-elm and op-knn: hannan-quinn criterion," in *European Symposium on Artificial Neural Networks (ESANN 2009)*, pp. 177–182, 2009.

[121] K. Neumann and J. J. Steil, "Batch intrinsic plasticity for extreme learning machines," in *International Conference on Artificial Neural Networks (ICANN 2011), Part I*, vol. 6791 of *Lecture Notes in Computer Science*, pp. 339–346, 2011.

[122] M. van Heeswijk, Y. Miche, T. Lindh-Knuutila, P. Hilbers, T. Honkela, E. Oja, and A. Lendasse, "Adaptive ensemble models of extreme learning machines for time series prediction," in *International Conference on Artificial Neural Networks (ICANN 2009), Part II*, vol. 5769 of *Lecture Notes in Computer Science*, pp. 305–314, 2009.

[123] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of online learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, pp. 119–139, Aug. 1997.

[124] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1998.

[125] J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky, "Bayesian model averaging: a tutorial," *Statistical Science*, vol. 14, no. 4, pp. 382–417, 1999.

[126] B. E. Hansen and J. S. Racine, "Jackknife model averaging," *Journal of Econometrics*, vol. 167, no. 1, pp. 38–46, 2012.

[127] J. Kidson, "Relationship of new zealand daily and monthly weather patterns to synoptic weather types," *International Journal of Climatology*, vol. 14, pp. 723–737, 1994.

[128] C. Rossby *et al.*, "Relations between variations in the intensity of the zonal circulation of the atmosphere and displacements of the semipermanent centers of action," *Journal of Marine Research*, vol. 2, pp. 38–55, 1939.

[129] E. N. Lorenz, "Seasonal and irregular variations of the northern hemisphere seal-level pressure profile," *Journal of Meteorology*, vol. 8, no. 1, pp. 52–59, 1951.

[130] D. W. Thompson and J. M. Wallace, "The arctic oscillation signature in the wintertime geopotential height and temperature fields," *Geophysical Research Letters*, vol. 25, no. 9, pp. 1297–1300, 1998.

[131] J. W. Hurrell, "Decadal trends in the north atlantic oscillation: regional temperatures and precipitation," *Science*, vol. 269, no. 5224, pp. 676–679, 1995.

[132] N. C. Stenseth, G. Ottersen, J. W. Hurrell, A. Mysterud, M. Lima, K. Chan, N. G. Yoccoz, and B. Ådlandsvik, "Studying climate effects on ecology through the use of climate indices: the north atlantic oscillation, el niño southern oscillation and beyond," *Proceedings of the Royal Society of London. Series B: Biological Sciences*, vol. 270, no. 1529, pp. 2087–2096, 2003.

[133] A. Schepen, Q. Wang, and D. Robertson, "Evidence for using lagged climate indices to forecast australian seasonal rainfall," *Journal of Climate*, vol. 25, no. 4, pp. 1230–1246, 2012.

[134] H. von Storch, E. Zorita and U. Cubasch, "Downscaling of global climate change estimates to regional scales: an application to iberian rainfall in wintertime," *Journal of Climate*, vol. 6, no. 6, pp. 1161–1171, 1993.

[135] K. Steffen and J. Box, "Surface climatology of the greenland ice sheet: greenland climate network 1995–1999," *Journal of Geophysical Research: Atmospheres*, vol. 106, no. D24, pp. 33951–33964, 2001.

[136] O. M. Johannessen, K. Khvorostovsky, M. W. Miles, and L. P. Bobylev, "Recent ice-sheet growth in the interior of greenland," *Science*, vol. 310, no. 5750, pp. 1013–1016, 2005.

[137] D. B. Enfield, A. M. Mestas-Nuñez, and P. J. Trimble, "The atlantic multidecadal oscillation and its relation to rainfall and river flows in the continental us," *Geophysical Research Letters*, vol. 28, no. 10, pp. 2077–2080, 2001.

[138] P. Winsor, J. Rodhe, A. Omstedt, *et al.*, "Baltic sea ocean climate: an analysis of 100 yr of hydrographic data with focus on the freshwater budget," *Climate Research*, vol. 18, no. 1/2, pp. 5–15, 2001.

[139] N. Schneider, E. Di Lorenzo, and P. P. Niiler, "Salinity variations in the southern california current," *Journal of Physical Oceanography*, vol. 35, no. 8, pp. 1421–1436, 2005.

[140] M. Viitasalo, "Mesozooplankton of the gulf of finland and northern baltic proper a review of monitoring data," *Ophelia*, vol. 35, no. 2, pp. 147–168, 1992.

[141] W. Matthäus and H. Schinke, "Mean atmospheric circulation patterns associated with major baltic inflows," *Deutsche Hydrografische Zeitschrift*, vol. 46, no. 4, pp. 321–339, 1994.

[142] J. Dippner and A. Ikauniece, "Long-term zoobenthos variability in the gulf of riga in relation to climate variability," *Journal of Marine Systems*, vol. 30, pp. 155–164, 2001.

[143] K. Mann and K. Drinkwater, "Environmental influences on fish and shellfish production in the northwest atlantic," *Environmental Reviews*, vol. 2, no. 1, pp. 16–32, 1994.

[144] A. Mysterud, N. C. Stenseth, N. G. Yoccoz, G. Ottersen, and R. Langvatn, *The Response of Terrestrial Ecosystems to Climate Variability Associated with the North Atlantic Oscillation*, pp. 235–262. American Geophysical Union, 2013.

[145] D. Straile, D. M. Livingstone, G. A. Weyhenmeyer, and D. G. George, *The Response of Freshwater Ecosystems to Climate Variability Associated with the North Atlantic Oscillation*, pp. 263–279. American Geophysical Union, 2013.

[146] K. F. Drinkwater, A. Belgrano, A. Borja, A. Conversi, M. Edwards, C. H. Greene, G. Ottersen, A. J. Pershing, and H. Walker, *The Response of Marine Ecosystems to Climate Variability Associated with the North Atlantic Oscillation*, pp. 211–234. American Geophysical Union, 2013.

[147] I. Kröncke, J. W. Dippner, H. Heyen, and B. Zeiss, "Long-term changes in macrofaunal communities off norderney (east frisia, germany) in relation to climate variability," 1998.

[148] G. A. Becker and M. Pauly, "Sea surface temperature changes in the north sea and their causes," *ICES Journal of Marine Science*, vol. 3, pp. 887–898, 1996.

[149] J. W. Dippner, K. Junker, and I. Kröncke, "Biological regime shifts and changes in predictability," *Geophysical Research Letters*, vol. 37, no. 24, pp. n/a–n/a, 2010.

[150] E. Kalnay *et al.*, "The ncep/ncar 40-year reanalysis project," *Bulletin of the American Meteorological Society*, vol. 77, no. 3, pp. 437–471, 1996.

[151] J. W. Dippner, G. Kornilovs, and K. Junker, "A multivariate baltic sea environmental index," *Ambio*, vol. 41, no. 7, pp. 699–708, 2012.

Aalto-DD 138/2014

ISBN 978-952-60-5856-6
ISBN 978-952-60-5857-3 (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**