

Development of Robust Inferential Sensors

*Industrial Application of
Support Vector Machines for Regression*

CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN

Jordaan, Elsie M.

Development of robust inferential sensors : industrial application of support vector machines for regression / by Elsie M. Jordaan. - Eindhoven:

Technische Universiteit Eindhoven, 2002.

Proefontwerp. - ISBN 90-386-0582-X

NUR 984

Subject headings : machine learning / robust statistics / intelligent control systems / artificial intelligence

2000 Mathematics Subject Classification : 68T05, 62G08

Reproduction: Universiteitsdrukkerij TU Eindhoven

This work has been sponsored by The Dow Chemical Company.

Development of Robust Inferential Sensors

*Industrial Application of
Support Vector Machines for Regression*

PROEFONTWERP

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
Rector Magnificus, prof.dr. R. A. van Santen, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen
op woensdag 18 december 2002 om 16.00 uur

door

Elsie Maria Jordaan

geboren te Groblersdal, Zuid-Afrika

De documentatie van het proefontwerp is goedgekeurd door de promotoren

prof.dr. J. Wessels
en
prof.dr. E.H.L. Aarts

Copromotor:
dr. G. F. Smits

*Aan my ouers,
wat my van God geleer het.*

Summary

The aim of the research was to investigate and further develop the technology for building and operating robust inferential sensors. Inferential sensors are mathematical models used to infer or predict the outcome of processes. Since the processes are not static, the inferential sensor needs to be adaptive to changing conditions in order to remain valid for an extended period of time. The technology needed for developing an adaptive inferential sensor requires the integration of its operational and application requirements into modelling software. Therefore, as a first step in this research project the design requirements of the inferential sensor first had to be defined. The following design requirements were identified: (1) complexity control; (2) ability to use high-dimensional data; (3) robustness; (4) generalisation ability; (5) data compression and outlier detection; (6) incorporation of prior knowledge; (7) self-diagnostic capability and (8) adaptive behaviour.

A new kind of learning machine, the Support Vector Machine (SVM), is used, since it best meets the first three design requirements. These three design requirements characterise the learning requirements of inferential sensors and are discussed in Part I of the design thesis. In the design thesis we focus on regression problems, as regression is one of the most widespread applications encountered in the chemical industry.

The design requirements (4), (5) and (6) are associated with the application stability requirements, which are discussed in Part II of the design thesis. In this part a new type of kernel is introduced. It is a mixture of a radial basis function and a polynomial kernel, which improves the generalisation ability of the inferential sensor model built on SVM technology. The improved generalisation ability enables the incorporation of certain types of prior knowledge readily available in the chemical industry. In our implementation of the data compression and outlier detection applications, we use new model-based approaches for detecting outliers as well as redundancy.

In Part III of the design thesis, which is concerned with the operational requirements, the last two design requirements, (7) and (8), are discussed. We investigated possible ways to evaluate the performance of the inferential sensor. In order to monitor and diagnose its own performance the inferential sensor needs, apart from the standard error statistics, also a kind of uncertainty level associated with its predictions. In the design thesis we therefore give an overview of various error statistics for measuring the overall performance as well as discuss a number of possibilities for determining an uncertainty level of a given prediction. We also discuss various levels of adaptation that can be implemented and present a support vector based approach for novelty detection as well.

For the development of robust inferential sensors it is required that the parameters of the learning machine can easily be estimated or determined. In Part I of the design thesis a heuristic is derived for estimating the regularisation parameter used by SVM for regression. Additionally, algorithmic pseudocodes are discussed for the optimisation of the parameters are given in the chapters where these parameters.

Finally, in Part IV of the design thesis a software tool that combines the SVM as learning algorithm with the design requirements is presented. A number of design requirements, like outlier and redundancy detection, are implemented in the form of application tools. Naturally, the algorithms for optimising the SVM parameters are implemented as well. The software tool further has a user interface that assists the user in setting or optimising the parameters and choices. It allows the user to investigate, analyse and interpret the results obtained. The software has been used at The Dow Chemical Company for some time and examples of these applications are given as well.

Contents

Summary	i
1 Introduction	1
1.1 Inferential sensors in the chemical industry	1
1.2 Design requirements	3
1.3 Project definition	7
1.4 Thesis layout in relation to the design	7
I Learning Requirements	11
2 Complexity Control	13
2.1 Introduction	13
2.2 Learning from data	15
2.2.1 The empirical risk minimisation principle	17
2.2.2 Bounds on the generalisation error	20
2.2.3 Structural risk minimisation principle	23
2.3 Support vector machines	25
2.3.1 Classification case	26
2.3.2 Regression case	32
2.4 Choice of complexity parameters	39
2.4.1 Size of the insensitive zone (ϵ)	39
2.4.2 Ratio of support vectors (ν)	41
2.5 Conclusion	42
3 High-dimensional Data and Spaces	45
3.1 Introduction	45
3.2 Curse of dimensionality	46
3.3 Kernel functions	50
3.4 Types of kernels	51
3.5 Choice of kernel and kernel parameters	55
3.5.1 Radial Basis Function kernel	55
3.5.2 Polynomial kernel	58
3.6 Conclusion	59

4	Robustness	61
4.1	Introduction	61
4.2	Linear ϵ -SVM for regression	62
4.3	Quadratic ϵ -SVM for regression	63
4.4	Choice of regularisation parameter	67
4.4.1	Results from the L-curve method	68
4.4.2	Estimate for C	70
4.4.3	Experimental results	74
4.5	Conclusion	76
II	Application Stability Requirements	79
5	Generalisation Ability	81
5.1	Introduction	81
5.2	Interpolation and extrapolation	82
5.3	Mixed kernel approach	83
5.4	Industrial application	88
5.5	Conclusion	91
6	Data Compression and Outlier Detection	95
6.1	Introduction	95
6.2	Outlier detection	96
6.3	Data reduction	101
6.4	Industrial example	105
6.5	Conclusion	105
7	Incorporate Prior Knowledge	109
7.1	Introduction	109
7.2	Boundary information	110
7.3	Multi-dimensional scaling	111
7.4	Variable ϵ	117
7.5	Conclusion	121
III	Operational Requirements	123
8	Adaptivity	125
8.1	Introduction	125
8.2	Novelty detection	126
8.3	Adaptation levels	127
8.4	Transduction	128
8.4.1	Euclidian distance	130
8.4.2	Delaunay tessellations	131
8.5	Conclusion	133

9 Self-Diagnostic Capabilities	135
9.1 Introduction	135
9.2 Error statistics	136
9.3 Classical confidence	141
9.4 Error bar estimation	143
9.5 Model disagreement	144
9.6 Conclusion	145
IV Implementation	147
10 Implementation of Software	149
10.1 Introduction	149
10.2 Applications of Part I	151
10.2.1 Complexity control	151
10.2.2 Handling high-dimensional data	153
10.2.3 Robustness	155
10.3 Applications of Part II	156
10.3.1 Generalisation ability	156
10.3.2 Data compression and outlier detection	156
10.3.3 Incorporating prior knowledge	156
10.4 Applications of Part III	157
10.4.1 Adaptivity	158
10.4.2 Self-diagnostic capabilities	158
10.5 Miscellaneous features	160
10.6 Conclusion	165
11 Conclusions	167
Bibliography	172
Samenvatting	181
Acknowledgements/Dankwoord	183
Curriculum Vitae	185

Chapter 1

Introduction

1.1 Inferential sensors in the chemical industry

Chemical and petrochemical plants increasingly rely on information from measurements in the plants to ensure quality of products and enable control [40]. There are mainly two types of chemical processes which are frequently used, namely continuous and batch processes [49]. Each type has its own issues to consider with respect to monitoring and control. However, one problem experienced by both types is that in many processes some process outputs are measured infrequently due to sampling limitations, which prevents early detection of disturbances in the process [100]. The early and accurate detection of the operational failure of a process plant and its associated instrumentation and control manipulators are of increasing industrial importance. Off-specification processes can result in poor product quality, lead to plant shutdowns, environmental contamination, or even be a hazard to human life [40].

If off-specification processes are detected quickly and accurately, the gains in productivity can be enormous [26]. Therefore, there is a need for real-time analysis in process plants. Real-time information from online analysers is the basis for effective process monitoring. Hardware online analysers are able to measure a wide variety of properties. However, there are many instances where hardware analysers are impractical due to costs, corrosion, maintenance, or other factors. Inferential analysers are therefore used to substitute hardware analysers. These inferential analysers are sometimes called *soft(ware) sensors*, *inferential sensors* or *virtual online analysers* [40]. They are in fact mathematical models implemented in computer software to calculate the analyser value or laboratory value as a function of process variables. Inferential sensors therefore often replace a hardware sensor and can be used for investigating what-if scenarios, alarm handling or optimisation of processes.

The process of constructing these mathematical models can take many forms. If possible these models are derived from first principle laws. But more often these models have to be derived from empirical data obtained from the production process, a pilot plant or the laboratory [40]. The models derived from empirical data can take two forms. One is a closed form formula in terms of the process variables. The classic statistical methods like (nonlinear) least squares (LS) and ridge regression are

examples of closed form formula empirical models [71],[11]. The other form is data-driven models which are expressed in terms of the observations. These models are often called black-box methods since the expression of the approximating function is not easily understandable in terms of the process variables [14]. Neural Networks (NN) and Support Vector Machines (SVM) are examples of data-driven models.

Inferential sensing using NNs has been applied successfully to thousands of applications worldwide in all areas of manufacturing and has therefore become the object of serious industrial research over the past decade. Three examples of vendors of commercial software for developing inferential sensors are Pavilion Technology(*Process Insights*) [53], Aspentech(*IQModel*) [2],[64] and Fisher-Rosemount Systems(*Intelligent Sensor ToolKit - ISTK*)[57]. However, in practice at The Dow Chemical Company, the inferential sensors developed using the commercial software showed several deficiencies with respect to robustness, more specifically complexity control, generalisation capabilities and long-term reliability [85]. investigate the process of building these inferential models.

The process of constructing data-driven models is called *machine learning*. Machine learning is part of a larger field of methods for the estimation of dependencies from data [11],[39]. The specific method of learning (or learning machine) used is one of the most important choices to make since the type of learning machine determines the characteristics of the resulting data-driven model. Each learning machine has certain characteristics that influence the overall performance of the inferential sensor. Most of the work being done in inferential sensor development is therefore concentrated on the learning part. However, inferential sensor development has many more issues to consider apart from the modelling process.

A typical project to develop a inferential sensor would involve a series of steps like data analysis, data reduction, development of transforms, model building, model explanation, implementation of the sensor and operation of the inferential sensor within the plant [40]. If this whole process is worked out and completed successfully, there is still one major aspect that limits the lifetime of a typical inferential sensor: The inability to adapt to changes [112]. Almost none of the processes being modelled is static or has been sampled over the full operating range before the development of the inferential sensor. This usually leads to a slow degradation of the performance of the inferential sensor over time and may eventually lead to its dismissal by the plant engineers. One solution could be that there is an extensive maintenance program to rebuild these inferential sensors whenever necessary [112]. However, this approach only works when there is a limited number of inferential sensors in operation at any given time. A significant scale-up in the number of inferential sensors to several hundreds would rapidly lead to an unmanageable situation.

Probably the only way to implement high-volume inferential sensors in any environment is if the following two concepts are developed: Firstly, the technology to build more intelligence into the inferential sensor itself, so that the system is able to diagnose its own performance. Secondly, the ability of the inferential sensor to become adaptive with respect to the changing environment. These two aspects are not addressed to any significant extent by the vendors of any of the packages that support inferential sensor development at this time. Furthermore, the inferential sensor not only has to perform outlier detection and novelty detection, but also use a learning

machine that has the ability to control the complexity of the model. All of these will increase the effectiveness of the inferential sensor to predict future measurements accurately.

The long term impact of using adaptive and intelligent inferential sensors is twofold. Not only will the production of various processes in chemical plants be improved tremendously, but the cost of maintenance of currently used inferential sensors will be cut as well.

In a previous study [33], a new type of learning machine, called the Support Vector Machine (SVM) [106], has been investigated and it was found that it exhibited many of the mentioned properties and characteristic required for the adaptive and intelligent inferential sensor. The research therefore focussed on the SVM method and also engineered a number of adaptations to the existing method in order to meet several other requirements.

On a higher level, we observed that the nature of the various properties and requirements is such that they can be put forward as a set of design requirements necessary for the development of an adaptive, intelligent, online inferential sensor. Therefore, in the light of this being a *design* thesis, the research is presented in terms of the design requirements.

Several of the design requirements have also been implemented together with the SVM into a software tool. This software tool is currently being used by research engineers of The Dow Chemical Company for the construction of inferential sensors [41]. The software has also been used in various projects for data analysis and materials design. The design thesis will also present a number of examples showing these industrial applications.

1.2 Design requirements

In the previous section several properties and characteristics of a inferential sensor were mentioned. The nature of these properties and characteristics are such that they can be used to describe design requirements for the inferential sensor. In this section a number of design requirements is given and the properties and characteristics of the inferential sensor that are associated with each instance are briefly explained.

Complexity control

Inferential sensors are often prone to underfitting or overfitting the learning data [22]. Underfitting occurs when the model fails to capture all variability in the data. Overfitting is exactly the opposite: The model also fits the noise present in the data. In both cases, the inferential sensor will have a sub-optimal ability to predict on the long run new, unseen data points. This may seriously affect its reliability as well as its lifespan.

The root cause of underfitting and overfitting can be explained in terms of the complexity of the model. A model's inherent complexity determines how much variability in the data it can account for. If too much complexity is used, the variability in the data due to noise is modelled as well. Thus, overfitting occurs. If the complexity

is too low and the model fails to account for the true variability, it is said to underfit [14],[11].

It seems straightforward that the solution to the problem is to choose the model from a set of functions that have the right complexity. However, it is seldom known what the true complexity of the functions should be [106],[11]. In classical modelling the set of functions is defined *a priori* without knowing whether the set of functions actually possesses the right level of complexity. Many methods therefore try to overcome this problem by adjusting the complexity recursively until an appropriate level of complexity is found. There are however no guarantees that the complexity is in fact optimal. Finally, even if the complexity level is known there is often no way of controlling the complexity during modelling.

In the context of inferential sensor development two things are required with respect to complexity, i.e. automatic optimisation and control.

Using high-dimensional data and spaces

In the past ten to fifteen years, the number of variables measured in a process plant has increased dramatically. The reason is that the industry began to understand that in order to improve their processes, they need to know exactly what has an influence on it. Currently it is not rare for scientists to receive data sets with hundreds of variables [39].

Since most of the problems encountered in real-life are nonlinear, linear learning machines map the input data into a higher dimensional space where the learning capability is increased [39],[11]. This higher dimensional space is also called the *feature space*. However, as the number of dimensions grows the dimensionality of the feature space can become computationally unmanageable. Furthermore, with an increasing number of dimensions the number of data samples needed for a sufficiently high sampling density increases exponentially [11]. A high sampling density is necessary to ensure that the learning machine can model more complex functions well. The phenomenon that the learning machine's computational and predictive performance can degrade as the number of input dimensions increases, is often called *the curse of dimensionality* [11],[10],[106].

One could seek to bypass the curse of dimensionality through dimensionality reduction methods. However, even after such measures are taken industrial data sets may still contain too many dimensions for classical learning machines. Therefore, the inferential sensors used in industry are required to use not only high-dimensional data sets but also overcome the curse of dimensionality.

Robustness

Inferential sensors often have to operate with data that contain noise and several outliers, and with the possibility that all kinds of changes may occur in the plant. The noise present in data sets obtained from industry are often such that it is neither constant nor normally distributed. Furthermore, the presence of outliers is often not known beforehand and they are often not easy to identify. In other words, industrial data sets are most of the time messy.

In the development of a inferential sensor, one has to take the noise into account as well as keep in mind that the noise is rarely constant over the whole input space. With respect to outliers: There are two ways of dealing with outliers, namely remove them or use techniques that are insensitive to them [11]. As many outlier detection algorithms do not work well with high-dimensional data sets, the presence of some outliers in the data cannot be ruled out [1]. It is very important that decisions about the process are not made based on the information conveyed by outliers [68]. The inferential sensor therefore has to be made insensitive to outliers. The term *robustness* is often used in industry to describe the inferential sensor's sensitivity to perturbations in the variables, parameters or learning data [22],[68].

One characteristic aspect of any inferential sensor is that it should not only make accurate predictions, but also be robust to moderate changes in the data [22]. Therefore the learning machine is required to resolve the subtle trade-off between accuracy and robustness.

Good generalisation capabilities

Although many measurements are being taken in a process, a process in a pilot plant cannot be run over the whole range of possible process conditions to obtain information over the whole input space. It is too expensive and very time consuming. The result is that the learning data often cover only a small part of the input space.

Therefore, when a process is in operation on the plant it may venture into operating regions that were unknown at the time of modelling. Most empirical models, such as NN's, do not extrapolate well [70]. In many inferential sensor applications efforts are made to restrict the inferential sensor's predictions to the known input space [70]. However, it is often expected by the process engineers that the model is able to predict unseen data within a reasonable distance from the known input space well. And for unseen data that are too far away, a "graceful degradation" of the model is preferable. That means, the model does not become instable and exhibit erratic behaviour.

The requirement for the inferential sensor is therefore that it is able to predict unseen data in regions of low data density in the known input space as well as regions that are outside the known input space.

Performing data compression and outlier detection

The quantity of observations taken in plants has increased tremendously over the past decade. In contrast to fifteen years ago, when there were never enough data, the size of data sets today is sometimes unmanageable. Compressing data sets with minimum loss of information is an essential capability for modern modelling tools [22],[14]. Data compression involves the detection and removal of redundant information. That is information that already exists and is only duplicate [14].

The other task under consideration is that of detecting and removing faulty measurements, often called outliers. In the design requirement on robustness we gave reasons for making the inferential sensor insensitive to outliers. Thus it would seem that outlier detection is not required anymore. However, outliers often contain useful information on abnormal behaviour of the process described by the data [1]. Con-

sequently, outlier detection still needs to be done in order to fully understand the behaviour of the processes under consideration.

Applications such as redundancy detection and outlier detection are therefore important requirements in the development of inferential sensors.

Incorporating prior knowledge

As processes get more complicated and more research is being done on these processes, more information about the physical laws, constraints and conditions is becoming available. It is to be expected that when prior knowledge is included into the learning process, the resulting model will have better generalisation abilities. The inferential sensor built on such a model may be more intelligent and should be more reliable [40].

It is therefore required that the new generation of inferential sensors should not only be built on empirical data but also try to incorporate any other form of information that is available.

Adaptivity

Due to the dynamic behaviour of many of the processes involved in industry, the inferential sensor needs to be retrained regularly. This leads to high maintenance costs and limited use of inferential sensors in industrial processes [40]. To increase the lifespan of the inferential sensor and reduce the costs, the inferential sensor therefore has to become adaptive to the changing conditions [112].

However, in order to adapt a inferential sensor to new information or conditions, the inferential sensor first needs to know when something novel has occurred. It is not a question of recognising obvious changes in the process like new equipment or procedures, but rather one of detecting subtle changes like e.g. seasonal behaviour or the slow degradation of a catalyst [112].

The final requirement is that the inferential sensor to be able to perform novelty detection and implement a procedure to adapt the model to the changed conditions which were detected as novel information.

Self-diagnostic capabilities

In industry, inferential sensors are often used for monitoring the performance of a process [70]. Therefore, it is necessary that the inferential sensor supplies the process engineer with information about the accuracy of its predictions. The inferential sensor thus monitors its own performance [22]. This requires that more intelligence is built into the inferential sensor so that a process engineer is warned early enough that the current model is not applicable anymore to the current process conditions. Thus reducing the manufacturing of off-specification products and preventing a false sense of trust.

The requirement for the inferential sensor is that it should have self-diagnostic capabilities in order to evaluate its own reliability.

1.3 Project definition

The main goal of the research project is to develop a framework for building and operating adaptive, intelligent inferential sensors on the basis of SVM technology.

This short description implies a lot. The character of the research is not only to do fundamental research but also to present a practical methodology that should make it easier for the user to construct an inferential sensor that meets a number of design requirements, exhibits certain characteristics and has the ability to perform various tasks. The research further investigated a promising new learning machine, the SVM (in particular the SVM for regression problems), to determine whether the resulting inferential sensor has the potential of satisfying the requirements that are imposed.

Several of the activities necessary to reach the goal are clearly of the research type. For example, the SVM for regression is still in its infancy and the majority of the research currently being done is for classification applications. Therefore, many of the results are only applicable for classification problems. Extending these results to regression applications often turns out to be far more difficult than expected. Therefore, most of the research activities are aimed at unresolved issues of SVM for regression. However, the research should always be executed with the sole purpose of realizing a specific design requirement or applicability in inferential sensor development.

The other major activity was the implementation of the SVM technology into a software tool combined with a number of design requirements in the form of application tools. Further activities consisted of obtaining references of publications related to inferential sensor development and SVMs for regression as well as constructing a data base for future queries.

General information on the design philosophy and its relation to the demands in industry can be found in a publication of the Stan Ackermans Institute (Centre of Technological Design) at the Eindhoven University of Technology [96].

1.4 Thesis layout in relation to the design

Each chapter of the thesis represents one of the design requirements as discussed in Section 1.2. The design requirements of the inferential sensor can be divided into three parts:

1. Learning Requirements

These requirements influence the most basic structure of the learning machine or modelling method to be used, namely the inductive principle implemented, possible loss-functions to be used and what kind of transformation is used. The design issues associated with this are

- complexity control
- high-dimensional spaces
- robustness

2. Application Stability Requirements

Here the requirements that affect the learning machine's performance are considered. These requirements either improve the behaviour of the resulting model

changing the learning method, or are considered to be used in pre-processing applications. The following design issues are associated with this criterion

- generalisation ability
- incorporating prior knowledge
- data compression

3. Operational Requirements

The requirements considered here are those that define the capabilities of the operational inferential sensor. These requirements are not part of the (off-line) learning process. The design issues are

- adaptivity
- self-diagnostic capabilities

Therefore, there are three main parts in the thesis. Part I involves the inherent learning requirements of the inferential sensor, Part II addresses the application stability requirements and Part III covers the operational requirements of the inferential sensor.

In Chapter 2 the first requirement, complexity control, is presented. The chapter discusses the SVM method and shows how complexity control is obtained by using the VC-dimension. The research was mainly concerned with the SVM for regression, since mainly regression applications are encountered in the chemical industry.

The chapter is followed by the second requirement which is about working with high-dimensional data. The chapter shows how SVMs use kernel functions to handle high-dimensional data efficiently and discusses how that enables the SVM to partially overcome the curse of dimensionality. It also discusses the characteristics of two main types of kernel functions, namely local and global kernels.

The final chapter of Part I is Chapter 4 which covers issues in the requirement concerning robustness properties. For robustness it is required that the inferential sensor is insensitive to noise and outliers. Different types of loss functions can be used to control the effect of errors in the data. The first contribution of the research, also presented in this chapter, is a fast and effective way to estimate the regularisation parameter which controls the trade-off between the error from the loss-function and the complexity.

Part II starts with Chapter 5, that deals with the generalisation capability. In this chapter industrial implications of generalisation are shown. Another contribution of the research, that is the engineering of a new kind of kernel which improves the extrapolation ability of the SVM, is also presented in the chapter.

The data compression design issue is discussed in Chapter 6. In this chapter an application which is based on the characteristics of the support vectors obtained by the SVM, is used to compress a data set by either removing redundant information or discarding outliers.

The chapter is followed by the design issue which involves the incorporation of prior knowledge. Here two contributions from the research can be found, i.e. the use of boundary information and the use of multi-dimensional scaling. The chapter also

investigates the practical use of another form of prior knowledge presented recently by [78]. This chapter concludes Part II.

Chapter 8 on the adaptive capabilities of a inferential sensor is one of the design issues discussed in Part III on the operational requirements. One of the main ingredients for adaptivity is the ability to detect novel information. A particular form of adaptivity is the transductive approach, which is discussed and illustrated in detail. The chapter finally proposes possible adaptation levels.

Part III aptly concludes with Chapter 9 on the self-diagnostic capabilities. An overview of various standard error statistics used to evaluate the overall performance of the inferential sensor including a measure from statistical learning theory (SLT) is given. In addition, the problems associated with the standard approach of constructing confidence limits are discussed and we briefly visit one of the most recent advances made in assigning uncertainty to predictions made by SVMs, i.e. the error bar estimation [20]. Finally, a possible non-statistical alternative, a model reliability measure used by NNs as well as genetic programming (GP), is considered [83].

Part IV contains a chapter which gives an overview of the software tool in which a large number of the design requirements and results obtained from the research is implemented. An additional feature is a complete user interface for setting and optimising parameters, selecting optimisation types and scaling choices, and displaying and analysing the performance of SVMs.

In the final chapter of the thesis, the conclusions are presented. In Figure 1.1 the relationships between the chapters are shown. The schematic overview also provides information about which chapters are pre-requisites for a specific chapter.

Note that in Chapters 6 to 8 the nature of the thesis is somewhat different than in the preceding chapters. Until Chapter 5, the proposed methods are meticulously tested in simulations and their merits and disadvantages are discussed. From Chapter 6 onwards, the thesis is more of an exploratory nature. We show possible applications as well as indicate various areas for future research. However, we feel that these exploratory chapters still need to be present in the thesis in order to give a full overview of the issues involved with the design requirements of inferential sensors.

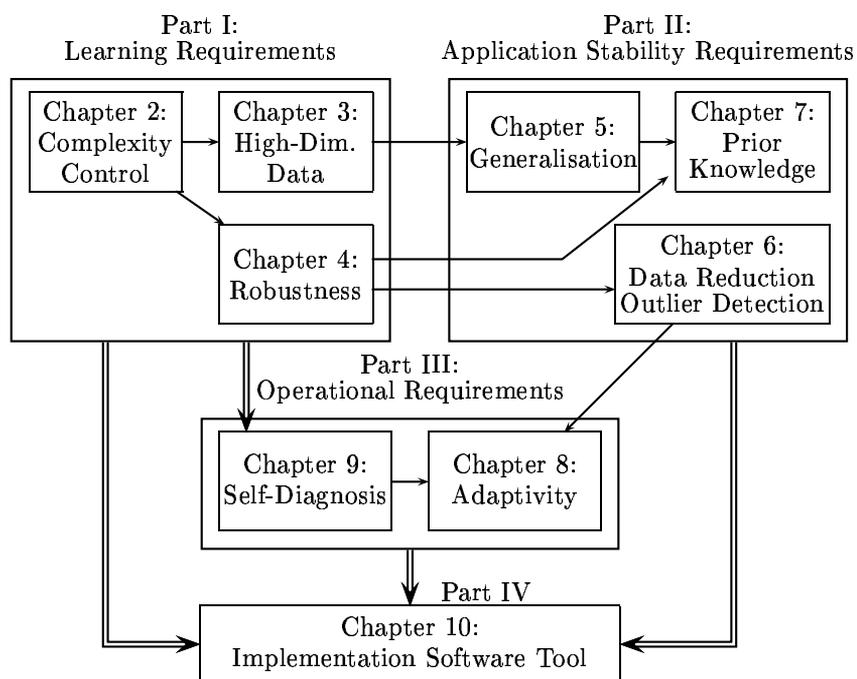


Figure 1.1: Relationship between chapters in the thesis.

Part I

Learning Requirements

Chapter 2

Complexity Control

2.1 Introduction

Inferential sensors often suffer from underfitting or overfitting the learning data [14]. Underfitting occurs when the model used has not the capacity to explain all variability in the data. Overfitting is exactly the opposite: the model has so much capacity that it also fits the noise present in the data. The root cause of underfitting and overfitting is explained in terms of the complexity of the model [11],[106],[39]. A model's inherent complexity determines how much variability in the data it can account for. If too much complexity is used, the variability in the data due to noise is modelled as well. Thus, overfitting occurs. If the complexity is too low and the model fails to account for the true variability, it is said to underfit.

Unfortunately, scientists seldom know what the optimal complexity of a model is or how to control it during the modelling phase. Therefore, there is a need for a inferential sensor that uses the optimal complexity for the given data set and has the ability to control it [22]. This is perhaps one of the most important design requirements a reliable inferential sensor has to meet, since it is known that an optimal complexity is a pre-requisite to good generalisation properties [106]. Under generalisation we understand the model's ability to predict unseen data within the known learning space ¹.

To address the problem of complexity one has to investigate the properties of the learning machine used in the modelling step [106],[11]. Many types of learning machines are constructed and currently used. The main difference between different types of learning machines lies in which inductive principle is used and how it is implemented [106]. The inductive principle tells us what to do with the learning data, i.e. how to enter the learning data into the learning machine. How to select the best model is a matter determined by the specific learning method that is used. Some well-

¹The design requirement Generalisation Ability discussed in Chapter 5 is broader than what is assumed here. This design requirement considers not only the interpolation ability of the inferential sensor, but also the extrapolation ability. Interpolation is defined as the ability to predict seen and unseen data within the known learning space. Under extrapolation we understand that all predictions are made outside the known learning space.

known inductive principles include regularisation [101], Structural Risk Minimisation (SRM)[108], Bayesian inference [66],[110] and Minimal Description Length (MDL) [72].

In the mid 1980's, when NNs were introduced, scientists observed that this new type of learning machine had not only a good learning capability but also had the potential of good generalisation abilities [102]. It was not until recently, that the reason was fully understood: NNs implemented the SRM principle instead of the Empirical Risk Minimisation principle (ERM) [106]. The SRM comes from a fairly complicated theory, called Statistical Learning Theory (SLT) which has been developed over the past thirty years by Vladimir Vapnik [108] and others [36],[37],[38],[104]. Since the publication of SLT results, the failure of many learning machines in small sample statistics was finally explained by their lack of complexity control. Where classical statistics deals successfully with large sample size problems, SLT is the first comprehensive theory built towards small sample learning problems. It has been shown that the inherent complexity of models based on empirical data is dependent on the sample size. Therefore, by taking into account the sample size, better solutions can be obtained than by just applying asymptotic results from classical statistics [106] [81].

It is important to note that as the number of dimensions in a data set increases, the number of observations should increase exponentially to represent the same density of information [11]. For example, in a five-dimensional data set a sample size of tens of thousands is still considered as small. In the chemical industry, where the number of dimensions in data sets has increased dramatically, scientists are dealing almost exclusively with small sample statistics.

As recent as 1993 a new kind of learning machine, called Support Vector Machine (SVM), was introduced into the artificial intelligence community [4],[105]. It makes use of the SRM principle. SVMs minimise the complexity of the model whilst maximising the learning capacity. This combined effect leads to a better generalisation ability of the model. The SVM method can be used for classification, regression estimation and density estimation [106]. Extensive research is currently being done for pattern recognition, which is a classification problem. Unfortunately, much less is being done in the area of regression estimation and density estimation. In the chemical industry, the majority of problems encountered are of the regression type and therefore our investigation focusses on the regression estimation problem.

In this chapter we give a literature review of issues involving complexity control. We first present the general setting of supervised learning tasks. Thereafter, the three main issues of SLT are discussed in short: Firstly the shortcomings of the ERM principle; Secondly, an alternative risk principle namely the SRM principle; And thirdly, a particular implementation of the SRM principle leading to a new kind of learning machine, the SVM, which can be used in industry for inferential sensor development.

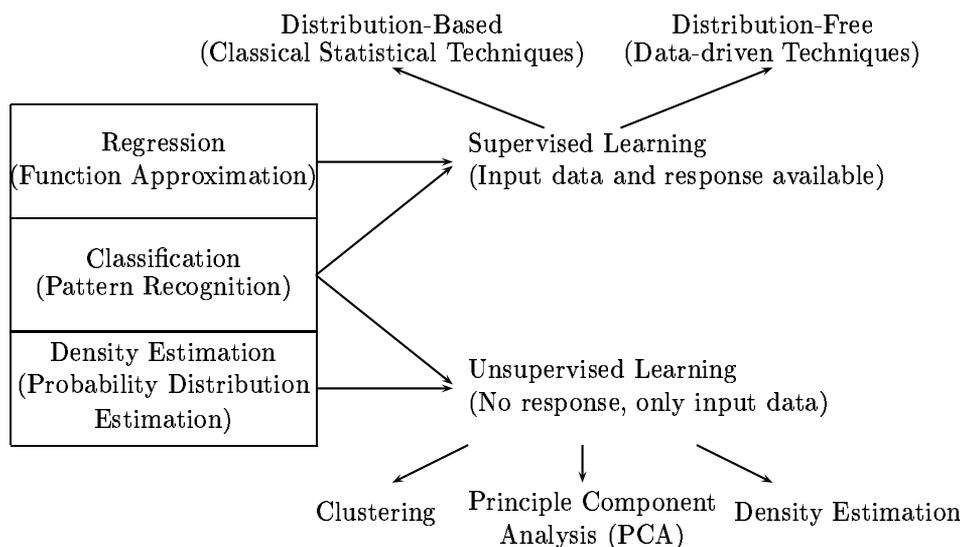


Figure 2.1: Schematic view of the general setting of learning tasks.

2.2 Learning from data

There are two ways to learn from data: *supervised* and *unsupervised* learning. For supervised learning a supervisor is available in the form of observed response data. Regression estimation and classification are typical problems solved through supervised learning. In unsupervised learning, no supervisor is available, thus no response data. Density estimation is a typical example of unsupervised learning. In Figure 2.1 an overview of the different learning problems and their connection to the type of learning is given.

In practice there is often no prior knowledge is available on the probability distribution models or probability density functions. This is especially the case in the chemical industry. In order to avoid making assumptions on the underlying distribution, one has to make use of distribution-free learning. Fortunately, in most cases response data are available and thus supervised learning can be applied. Note that some learning methods can be applied to both supervised and unsupervised learning. From here on we are only concerned with distribution-free supervised learning.

The supervised learning problem is the problem of finding a desired dependency or structure using a limited number of observations. The general model of learning from examples consists of three components:

1. A generator (G) generates vectors $\mathbf{x} \in \mathbb{R}^n$ from a fixed, unknown probability distribution function $F(\mathbf{x})$.
2. A supervisor (S) returns an output value $y \in \mathbb{R}$ to every input vector \mathbf{x} according to a conditional distribution function $F(y|\mathbf{x}), y \in Y$, also unknown.

3. A learning machine (LM) selects the best approximating function from a set of functions $f(\mathbf{x}, \alpha)$, $\alpha \in \Lambda$, where Λ is a set of parameters, on the basis of the given observations (\mathbf{x}_i, y_i) , $i = 1, \dots, \ell$.

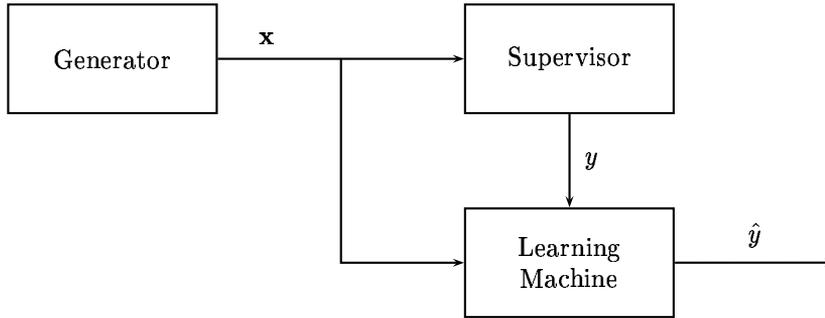


Figure 2.2: The General Learning Machine.

The relationships between the three components (G), (S) and (LM) are given Figure 2.2. The learning machine has to choose from the set $f(\mathbf{x}, \alpha)$, $\alpha \in \Lambda$, the function which best approximates the response of the supervisor. All learning machines basically map an input vector \mathbf{x} into a higher dimensional feature space U and then construct an approximating function in this features space.

The best approximating function is the function that has the lowest risk of making errors. That is, when the difference between the response y of the supervisor of a given input \mathbf{x} and the response $\hat{y} = f(\mathbf{x}, \alpha_0)$ of the learning machine is the lowest. This difference or discrepancy is determined by a so-called *Loss function*, $L(y, f(\mathbf{x}, \alpha))$. The goal is to minimise the expected value of the loss, given by

$$R(\alpha) = \int_{\mathbb{R}^n \times Y} L(y, f(\mathbf{x}, \alpha)) dF(\mathbf{x}, y). \quad (2.1)$$

Equation 2.1 is called the *Risk functional*. Note that the joint probability function $F(\mathbf{x}, y) = F(\mathbf{x})F(y|\mathbf{x})$ is unknown. The only information available is the training set of ℓ independent and identically distributed observations,

$$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell), \quad (2.2)$$

drawn according to $F(\mathbf{x}, y)$.

The difference between different types of learning machines is based on two features: Firstly, on which inductive principle is used; Secondly, on which type of loss function that is used. For example, the ERM inductive principle is used by least squares and maximum likelihood methods, whereas the L_2 -loss function defines a regression learning machine. Next, we state the two main learning problems for supervised learning: classification and regression estimation. Each learning problem requires the use of a different loss-function.

Classification

For classification problems the response y of the supervisor takes only discrete values. Every value identifies a certain class. Let y take only two values and consider the loss-function:

$$L(y, f(\mathbf{x}, \alpha)) = \begin{cases} 1 & \text{if } y = f(\mathbf{x}, \alpha) \\ -1 & \text{if } y \neq f(\mathbf{x}, \alpha). \end{cases} \quad (2.3)$$

The risk functional (2.1), using the loss-function in (2.3), determines the probability that the indicator function $f(\mathbf{x}, \alpha)$ gives a different answer than the supervisor does, that is to make a *classification error*. The learning problem for classification is therefore defined as finding the indicator function which minimises the probability of making classification errors when the probability measure $F(\mathbf{x}, y)$ is unknown, but the learning data in (2.2) are given.

Regression estimation

In the problem of regression estimation, the response y of the supervisor can be any real value. The functions $f(\mathbf{x}, \alpha)$, $\alpha \in \Lambda$, are also real valued and represent the potential regression function. A typical loss function that is used for regression problems is given by

$$L(y, f(\mathbf{x}, \alpha)) = (y - f(\mathbf{x}, \alpha))^2. \quad (2.4)$$

The risk functional that uses (2.4) as loss-function determines the prediction error of the regression function $f(\mathbf{x}, \alpha)$ with respect to the supervisor's response. Therefore, the learning problem for regression is that of finding the regression function which minimises the probability of making prediction errors when the probability measure $F(\mathbf{x}, y)$ is unknown, but the learning data in (2.2) are given.

2.2.1 The empirical risk minimisation principle

The risk functional $R(\alpha)$ in (2.1) cannot be minimised since the probability measure $F(\mathbf{x}, y)$ is unknown. The risk functional $R(\alpha)$, as it is stated in (2.1), is therefore only an a posteriori measure. In order to minimise the risk functional the unknown probability distribution $F(\mathbf{x}, y)$ is replaced by the empirical distribution, based in the learning data given in (2.2).

Using this inductive principle the risk functional in (2.1) is replaced by the *empirical risk functional*

$$R_{\text{emp}}(\alpha) = \frac{1}{\ell} \sum_{i=1}^{\ell} L(y_i, f(\mathbf{x}_i, \alpha)), \quad (2.5)$$

which is constructed on the basis of the given learning data set (\mathbf{x}_i, y_i) , $i = 1, \dots, \ell$.

The inductive principle applied in (2.5) is known as the Empirical Risk Minimisation (ERM) principle. The ERM is very important in learning theory. It is a principle which is used in classical methods like least squares and maximum likelihood [106],[11]. The different learning machines arise from using different types of

loss-functions. For example, the least squares method minimises

$$R_{\text{emp}}(\alpha) = \frac{1}{\ell} \sum_{i=1}^{\ell} (y_i - f(\mathbf{x}_i, \alpha))^2. \quad (2.6)$$

Consistency of the ERM principle

In order to have a successful learning process, it is required that the learning machine achieves a small value of actual risk. An important question has to be asked: When does a learning machine achieve a small value of actual risk and when does it not? At first, one would think that the smallest value of the empirical risk should be sufficient for achieving a small value of actual risk. Unfortunately, the answer is not that straightforward. A small value of the ERM does not necessarily imply a small value of actual risk. For that, the ERM principle needs to be consistent.

Consider a given set of identically and independently distributed (i.i.d.) learning data $z_i = (\mathbf{x}_i, y_i)$, $i = 1, \dots, \ell$. Let $Q(z, \alpha)$ be a function that minimises the empirical risk functional in (2.5), that is

$$R_{\text{emp}} = \sum_{i=1}^{\ell} Q(z_i, \alpha).$$

Definition 1. *The ERM principle is consistent for a set of functions $Q(z, \alpha)$, $\alpha \in \Lambda$, and the probability distribution function $F(z)$ if the following two sequences converge in probability to the same limit:*

$$R(\alpha_\ell) \xrightarrow{\ell \rightarrow \infty} \inf_{\alpha \in \Lambda} R(\alpha) \quad (2.7)$$

$$\text{and } R_{\text{emp}}(\alpha_\ell) \xrightarrow{\ell \rightarrow \infty} \inf_{\alpha \in \Lambda} R(\alpha). \quad (2.8)$$

This means that the ERM is consistent if both the expected risk (2.7) and the ERM (2.8) converge to the actual risk, that is the infimum of the expected risk, as seen in Figure 2.3. From the figure it is quite clear why learning machines based on the ERM principle have difficulties with small samples. The ERM will only converge to the actual risk, when the number of observations in the learning data tends to infinity.

To construct a learning machine that is consistent, even for small samples, we need to know the necessary and sufficient conditions for consistency. These are stated in the Key Theorem of Learning Theory. The proof of the theorem can be found in [108].

Theorem 1. Key Theorem of Learning Theory. *For bounded loss functions, the ERM principle is consistent if and only if the empirical risk converges uniformly to the actual risk in the following probabilistic sense:*

$$\lim_{\ell \rightarrow \infty} P \left[\sup_{\alpha \in \Lambda} |R(\alpha) - R_{\text{emp}}(\alpha)| > \epsilon \right] = 0, \forall \epsilon > 0. \quad (2.9)$$

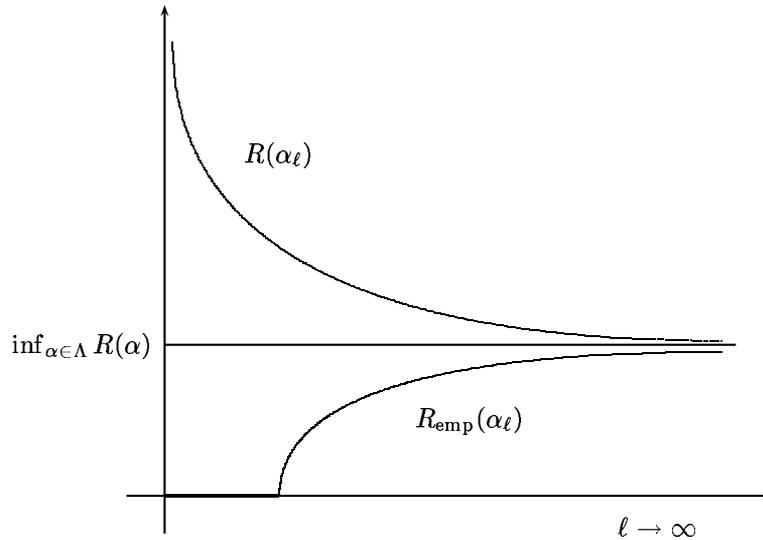


Figure 2.3: The consistency of the learning process

From the conceptual point of view, the theorem is extremely important for learning theory. It states that the necessary and sufficient conditions for consistency of the ERM are determined by the “worst” function in the set of possible functions. Thus, any analysis of the ERM must result in a “worst-case” analysis [106].

Unfortunately, the conditions (2.9) set in the Key Theorem of Learning Theory are not useful for constructing learning machines, since the bounds they pose are non-constructive as they cannot be implemented directly into learning algorithms. Fortunately, Statistical Learning Theory derived constructive bounds that are valid for any learning machine. These bounds are expressed in terms of the sample size l and the VC-dimension (Vapnik-Chervonenkis) h of the set of functions used by the learning machine. Furthermore, the VC-dimension provides us with conditions under which the ERM principle has a fast rate of convergence. Using the concept of VC-dimension the following holds true:

For a fast rate of convergence and distribution-free consistency of the ERM principle the necessary and sufficient condition is that the set of approximating functions implemented by the learning machine has a finite VC-dimension.

A full description of the conditions for consistency as well as the conditions for a fast rate of convergence of the ERM, can be found in two books by Vapnik [106],[108].

We will not discuss the concept of the VC-dimension in full here, since in the scope of the rest of the thesis, it is only necessary to know the following.

- The VC-dimension h is a measure of the richness or flexibility of the set of functions from which the model will be constructed. If the functions are rich

enough, the learning machine will have sufficient *capacity* to construct a model that has a suitable level of complexity for the data available [10].

- For both classification and regression problems, the VC-dimension of the loss-function $L(y, f(\mathbf{x}, \alpha))$ equals the VC-dimension of the approximating function $f(\mathbf{x}, \alpha)$ [11].
- The VC-dimension for linear indicator functions in a n -dimensional space (as is used in the loss-function of classification problems), is $h = n + 1$ [106].
- For the set of real functions in \mathbb{R}^n (as is used in the loss-function of regression problems), the VC-dimension is $h = n + 1$ [106].
- Generally speaking the VC-dimension of a set of functions does not coincide with the number of parameters of the set. It can be both larger than the number of parameters and smaller than the number of parameters. It is the VC-dimension, rather than the number of parameters, which is responsible for the generalisation ability of learning machines [106].

For the definitions of the VC-dimension for linear indicator functions and real-valued function, as well as a number of examples that illustrates the determination of the VC-dimension see [11].

In Section 2.2.3 an inductive principle which satisfies the conditions for consistency and has fast rate of convergence even for small samples, is discussed. First, we need to state constructive bounds on the generalisation.

2.2.2 Bounds on the generalisation error

SLT provides several non-constructive bounds that are conceptually important, but difficult to determine in learning algorithms. In order to construct learning machines that are able to control the complexity, we need constructive bounds on the generalisation error. The necessity of complexity control can be seen in Figure 2.4, where all the approximating functions have an empirical risk of zero, but differ in smoothness. Clearly in this case, the approximating function with highest smoothness will have better generalisation ability. Note that smoothness is associated with the complexity of the function [11],[39]. In most practical problems high smoothness indicates low complexity and vice versa [78]. Thus, smoothness is inversely related to the complexity.

Using the VC-dimension, SLT gives constructive bounds [108]. Furthermore, these bounds are distribution-free and therefore applicable to any learning machine. Next we state the bounds for bounded, non-negative functions and for unbounded functions, since they are used by classification and regression learning machines respectively.

Generalisation bounds for bounded, non-negative functions

For bounded, non-negative functions $0 \leq Q(z, \alpha) \leq B$ and given confidence level $(1 - \eta) > 0$, the bound for the generalisation ability is, with probability of at least

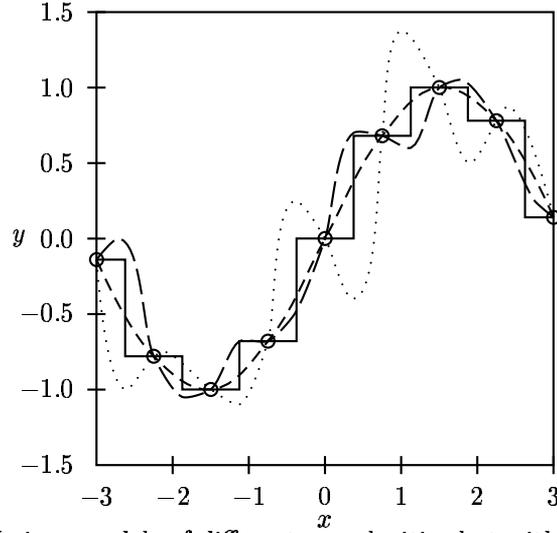


Figure 2.4: Various models of different complexities but with equal empirical risk.

$1 - \eta$,

$$R(\alpha_\ell) \leq R_{\text{emp}}(\alpha_\ell) + \frac{B\varepsilon}{2} \left(1 + \sqrt{1 + \frac{4R_{\text{emp}}(\alpha_\ell)}{B\varepsilon}} \right), \quad (2.10)$$

where

$$\varepsilon = 4 \frac{h(\ln(2\ell/h) + 1) - \ln(\eta/4)}{\ell}, \quad (2.11)$$

if the set of functions $Q(z, \alpha)$, $\alpha \in \Lambda$, contains a finite number of elements and has a finite VC-dimension h [108].

The bound in (2.10) becomes large when the *confidence level*, $1 - \eta$ is large. If $\eta \rightarrow 0$, then $\varepsilon \rightarrow \infty$ in (2.11) and the right-hand side of the bound grows large. This means that any estimate obtained from a finite number of samples, cannot have an arbitrarily high level of confidence. Hence, it is reasonable to determine the confidence level as a function of the number of samples. Vapnik recommended the following rule [108]

$$\eta = \min\left(\frac{4}{\sqrt{\ell}}, 1\right). \quad (2.12)$$

Thus the bound can be presented as

$$R(\alpha) \leq R_{\text{emp}}(\alpha) + \Phi\left(R_{\text{emp}}(\alpha), \frac{\ell}{h}, \frac{-\ln \eta}{4}\right), \quad (2.13)$$

where $\Phi(\cdot)$ is a scalar and called the *VC confidence interval*, because it estimates the difference between the empirical error and the actual error. It is important not to confuse the term *VC confidence interval* with the classical confidence interval. Note that the first term in (2.13) depends on a particular function from the set of functions whilst the second term mainly depends on the VC dimension [11],[106],[108].

An analysis of the behaviour of Φ as a function of the sample size ℓ with all other parameters fixed, shows that Φ mainly depends on \mathcal{E} , which monotonically decreases (to zero) with ℓ . In Figure 2.3, Φ corresponds to the upper bound on the distance between the two curves for any fixed ℓ [11]. \mathcal{E} also clearly shows the strong dependence of Φ on the ratio ℓ/h^2 . There are clearly two regimes. One where the sample size is small and finite which corresponds to a small ratio of ℓ/h . The other where the sample size is large, results in a large ratio. When the ratio is large, the value of the VC confidence interval becomes small. The empirical risk can then be used safely as a measure of the actual risk. However, for a small ratio (small sample sizes), the value of the VC confidence interval cannot be ignored.

Minimising the VC confidence interval not only ensures that the chosen model leads to a small risk of making estimation errors, but also comes from a function class with small complexity. In order to minimise the VC confidence interval, we need to make the VC dimension a controllable variable. The SRM principle provides a formal mechanism to achieve that.

Generalisation bounds for unbounded functions

The bounds on the generalisation error for regression problems need to be valid for unbounded loss-functions, since the bounds on the true function or the additive noise are not known. This means that there is always the probability of observing large response values which will lead to large, possibly unbounded, values for the loss-function. Therefore, strictly speaking, it is impossible to estimate the probability of such large responses based on the finite training error alone [11],[106],[108].

Fortunately, SLT provides us with some general characteristics on the distributions of unbounded loss-functions where large values of loss do not occur very often [106]. These characteristics describe the “tails of distributions”, i.e. the probability of observing large values of loss. For distributions with light tails, i.e. small probability of observing large values of loss, a fast rate of convergence is possible.

For the set of unbounded non-negative functions $0 \leq Q(z, \alpha) \leq B$ and given confidence level $(1 - \eta) > 0$, the bound for the generalisation ability is,

$$R(\alpha_\ell) \leq \frac{R_{\text{emp}}(\alpha_\ell)}{(1 + c\sqrt{\mathcal{E}})_+}, \quad (2.14)$$

where \mathcal{E} is defined by (2.11) if the set of functions $Q(z, \alpha), \alpha \in \Lambda$, contains a finite number of elements and has a finite VC-dimension h [108]. The constant c depends on the “tails of the distributions” of the loss-function. For most practical problems it is found to be $c = 1$ [11].

The bound in (2.14) provides an upper bound on the expected risk. As $\eta \rightarrow 0$ ($1 - \eta$ high), the value of $\mathcal{E} \rightarrow \infty$ when the other parameters are fixed. Therefore, the denominator in (2.14) equals zero and the bound would approach infinity. Again one can reason that any estimate from a finite number of samples cannot have an arbitrarily high level of confidence. Thus there is always a trade-off between the accuracy given by the bound and the degree of confidence in the bound. Furthermore,

²In many publications Φ is therefore expressed only in terms of ℓ/h .

in terms of the sample size ℓ , it is clear that when $\ell \rightarrow \infty$, the ratio ℓ/h is large and $\mathcal{E} \rightarrow 0$. The denominator in (2.14) then approaches one. Therefore, decreasing the value of the expected risk is a matter of decreasing the empirical risk. For large sample sizes the empirical risk can then be safely used. On the other hand, when the ratio ℓ/h is small, the denominator in (2.14) cannot be ignored. Note that the denominator is *not* referred to as the VC confidence interval. However, as with the VC confidence interval, there is a strong dependence on the ratio of ℓ/h and thus on the VC-dimension. Also note that the numerator in (2.14) depends on a particular function from the set of functions.

As in the case for classification, we need to make the VC-dimension a controllable variable in order to minimise the bound in (2.14). In the next section we argue that the SRM principle provides the means to do so.

2.2.3 Structural risk minimisation principle

Recall that the ERM principle is only intended for large sample sizes. That is when the ratio ℓ/h is large and $\mathcal{E} \approx 0$ in the bound for classification (2.10) or for regression (2.14). Then the empirical risk is close to the true risk and a small value of empirical risk guarantees a small true risk. However, when the ratio ℓ/h is small the VC confidence interval in (2.13) or both the numerator and denominator in (2.14) need to be minimised.

For small samples the set of functions used in the learning process should have the right level of complexity in order for the learning machine to be able to generalise well. Therefore, when constructing learning machines for small samples, we need to control the complexity of the resulting model. This is achieved by restricting the set of functions from which the approximating function is chosen, to functions that have a suitable learning *capacity* for the number of available learning data.

Under the SRM principle the set S of functions $Q(z, \alpha), \alpha \in \Lambda$ has a *structure* that consists of nested subsets of functions $S_k = \{Q(z, \alpha), \alpha \in \Lambda_k\}$ such that

$$S_1 \subset S_2 \subset \dots \subset S_k \subset \dots, \quad (2.15)$$

where each element of the structure S_k has a finite VC-dimension and should be either bounded or if unbounded should satisfy some general conditions³ to ensure that the risk functional does not grow unboundedly. By definition of the structure, the ordering of the elements, according to their VC-dimensions, is as follows

$$h_1 \leq h_2 \leq \dots \leq h_k \leq \dots \quad (2.16)$$

For a given set of ℓ samples the SRM chooses the function $Q(z, \alpha_k^*)$ that minimises the empirical risk for the functions in the subset S_k , for which the *guaranteed risk*, the combined terms in (2.13) for classification and in (2.14) for regression, is minimal.

³A set of nonnegative functions $Q(z, \alpha), \alpha \in \Lambda_k$ should satisfy

$$\sup_{\alpha \in \Lambda_k} \frac{\sqrt{E(Q^p(z, \alpha))}}{E(Q(z, \alpha))} \leq \tau_k < \infty.$$

See [108] for more information.

In both cases the SRM defines a trade-off between the quality of the approximation of the given learning data and the complexity of the approximating function. One could see the effect of the trade-off in Figure 2.5.

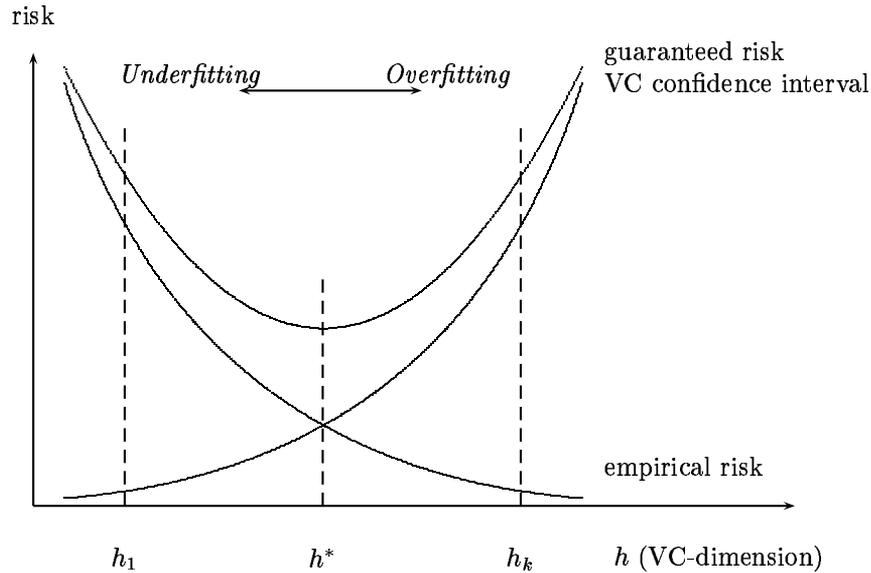


Figure 2.5: The structural risk minimisation principle.

In Figure 2.5 it is clear that the SRM principle is concerned with finding the right balance between the learning ability and the generalisation ability of the learning machine. If a too high complexity is used by the learning machine, the learning ability may be good, but the generalisation ability not. The learning machine will overfit the data. In Figure 2.5, we see that the region to the right of the optimal complexity h^* , corresponds with overfitting of the learning data. On the other hand, when the learning machine uses too little complexity, it may have a good generalisation ability, but not a good learning ability. This underfitting of the learning machine corresponds with the region left of the optimal complexity. The optimal complexity of the learning machine is the set of approximating functions with lowest VC-dimension and lowest training error.

Construction of learning machines using the SRM

The implementation of the SRM principle requires the *a priori* specification of the structure on the set of approximating (or loss) functions. For such a given set, the optimal model estimation amounts to the following two steps.

1. Select an element of the structure which has optimal complexity.
2. Estimate the best model of this element.

Therefore, unlike classical methods, learning machines that implement the SRM principle, provide analytical estimates for model selection based on the bounds for generalisation error.

There are two approaches of implementing the SRM inductive principle in learning machines:

1. Keep $\Phi(\cdot)$, the VC confidence interval fixed and minimise the empirical risk $R_{\text{emp}}(\alpha)$.
2. Keep the empirical risk, $R_{\text{emp}}(\alpha)$ fixed and minimise the VC confidence interval $\Phi(\cdot)$.

NN algorithms implement the first approach, since the number of hidden nodes is defined *a priori* and therefore the complexity of the structure is kept fixed.

The second approach is implemented by the SVM method where the empirical risk is either chosen to be zero or set to an *a priori* level (the value of the ϵ -insensitive zone) and the complexity of the structure is optimized. Note that we still have not provided the exact means for optimising the VC-dimension nor what kind of structure should be used. We will show in the next section that by using a specific format for the set of approximating functions and choice of structure, the SVM is able to minimise the VC-dimension and therefore the generalisation bounds.

2.3 Support vector machines

The second approach to implement the SRM is the support vector machine method, where the empirical risk is kept fixed while the VC confidence interval is minimised. In general the SVM maps the input data into a higher dimensional feature space. The mapping can be done nonlinearly and the transformation function is chosen *a priori*. Let us denote this transformation function with $\phi(\mathbf{x})$ for the time being. In Chapter 3 we will discuss these functions in more detail and introduce the concept of a kernel $K(\mathbf{x}, \hat{\mathbf{x}}) = \langle \phi(\mathbf{x}), \phi(\hat{\mathbf{x}}) \rangle$ ⁴. In the rest of this chapter \mathbf{x} is mainly used. Only during the final step of deriving the SVM it is replaced with the transformation $\phi(\mathbf{x})$.

In the feature space the SVM finally constructs an optimal approximating function which is linear in its parameters

$$f_{\mathbf{w},b}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b, \quad (2.17)$$

where \mathbf{w} and b have to be determined. In the classification case, $f_{\mathbf{w},b}(\mathbf{x}) = 0$ is called a decision function or *hyperplane*, and the optimal function is called an optimal separating hyperplane. In the regression case, $f_{\mathbf{w},b}(\mathbf{x})$ is called an approximating function or in statistical terms a hypothesis. In Figure 2.6, the general scheme of the SVM is given.

Now let us provide a structure on the set of approximating functions for which the VC-dimension is minimised. Let $\|\mathbf{w}\|$ define a structure on the set of approximating functions

$$f(\mathbf{x}, \alpha) = \langle \mathbf{w}, \mathbf{x} \rangle + b, \quad \alpha \in \Lambda, \quad (2.18)$$

⁴The notation $\langle \mathbf{x}, \mathbf{z} \rangle$ defines the inner product between vectors \mathbf{x} and \mathbf{z} .

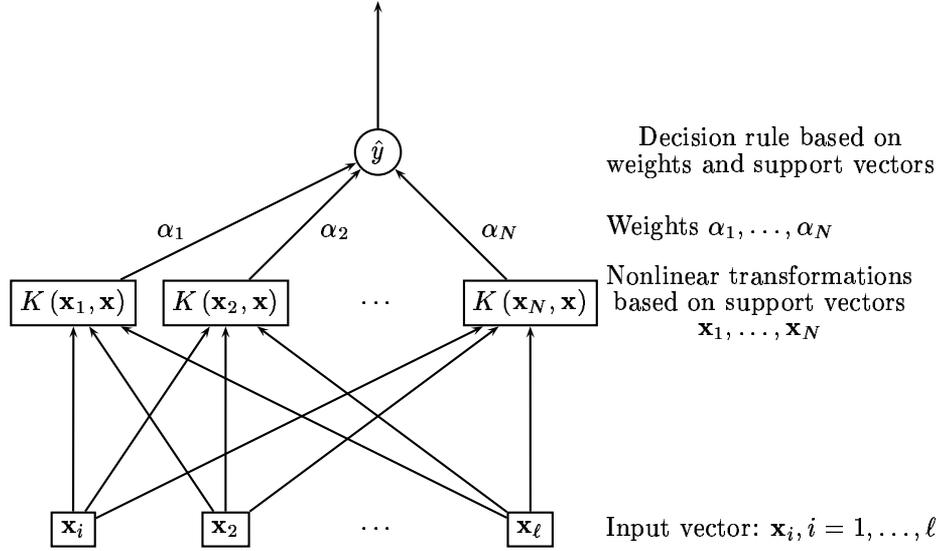


Figure 2.6: The general scheme of the SVM.

such that the elements of the set S_A are analysed using $\|\mathbf{w}\| \leq A$. Then, if $A_1 \leq A_2 \leq A_3 \leq \dots \leq A_n$, the set S_A can be nested such that $S_{A_1} \subset S_{A_2} \subset S_{A_3} \subset \dots \subset S_{A_n}$. Vapnik proved in [108] that the VC-dimension h of a set canonical hyperplanes in \mathbb{R}^n such that $\|\mathbf{w}\| \leq A$ is

$$h = \min(R^2 A^2, n) + 1, \quad (2.19)$$

where all the training data points (vectors) are enclosed by a sphere of the smallest radius R .

Similarly, for a set of real valued functions in \mathbb{R}^n such that $\|\mathbf{w}\| \leq A$, the VC-dimension is

$$h = \min\left(\frac{R^2}{\rho^2} + 1, n\right), \quad (2.20)$$

where ρ is a function of $\mathbf{w}/\|\mathbf{w}\|$ and all the training data points (vectors) are enclosed by a sphere of the smallest radius R .

Therefore, minimisation of $\|\mathbf{w}\|$ is an implementation of the SRM principle because a small value of $\|\mathbf{w}\|$ will result in a small value of h and the VC-dimension h is consequently minimised.

In the next two sections we will present the implementation of the Support Vector Machine for classification and regression where $\|\mathbf{w}\|$ is minimised, and the R_{emp} is either zero or kept fixed at a certain level.

2.3.1 Classification case

Consider the following collection of learning data

$$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell), \mathbf{x}_i \in \mathbb{R}^n, y_i \in \{-1, 1\}. \quad (2.21)$$

In the classification problems we want to find a decision rule $D(\mathbf{x}) = (\mathbf{w} \cdot \mathbf{x}) + b$ that classifies without error the learning data in (2.21) into the two classes $\{\mathbf{x} | D(\mathbf{x}) = 1\}$ and $\{\mathbf{x} | D(\mathbf{x}) = -1\}$, based on the data given in (2.21). It is said that such a decision rule, also called *hyperplane*, separates the data set without error. At this moment we assume that the data are separable. We will briefly discuss the non-separable case later. There exist many hyperplanes that separate the data, but we are looking for the one that is *optimal*.

Optimal separating hyperplane

The optimal separating hyperplane not only separates the data without error, it also has the largest distance between the closest vectors to either side of the hyperplane, as seen in Figure 2.7. This distance is called the *margin*.

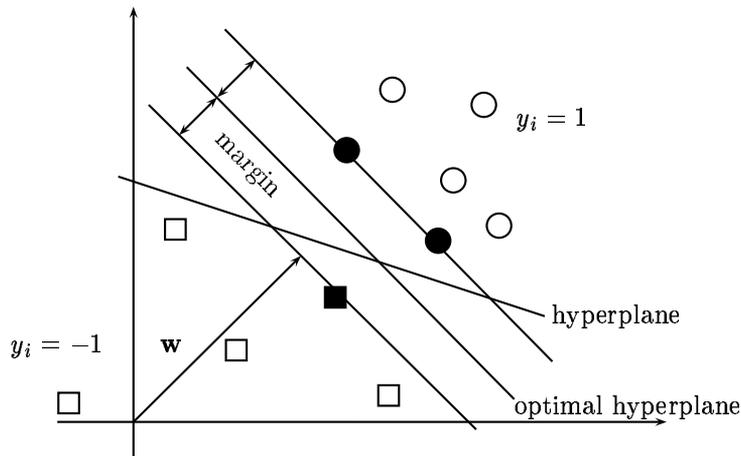


Figure 2.7: The optimal separating hyperplane.

The maximum margin is achieved by choosing the hyperplane with the smallest norm of coefficients. This has the implication that the smaller the value of $\|\mathbf{w}\|$ the larger is the value of the margin τ . Recall that it was argued that minimising $\|\mathbf{w}\|$ results in minimising the VC-dimension. Therefore, the optimisation problem which implements the SRM principle can be stated as

$$\text{minimise } \|\mathbf{w}\|, \quad (2.22a)$$

$$\text{subject to } y_i[(\mathbf{w}, \mathbf{x}_i) + b] \geq 1, \quad i = 1, \dots, \ell. \quad (2.22b)$$

However, minimising the norm of \mathbf{w} results in minimizing a square root which is difficult to solve. Since the square root function is a monotonic function, one can minimise the squared norm and reach the same result. Furthermore, the squared norm is used in (2.19) which gives a bound on the VC-dimension. Finally, through

this slight change in the objective function, the optimisation problem becomes a quadratic programming (QP) problem with convex constraints:

$$\text{minimise } \frac{1}{2} \|\mathbf{w}\|^2, \quad (2.23a)$$

$$\text{subject to } y_i [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] \geq 1, \quad i = 1, \dots, \ell. \quad (2.23b)$$

The solution of a QP problem is unique, which is a major advantage over NNs that have local minima.

A QP problem is solved by finding the saddle point of the Lagrange functional,

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^{\ell} \alpha_i ([\langle \mathbf{w}, \mathbf{x}_i \rangle - b] y_i - 1), \quad (2.24)$$

where α_i are the Lagrange multipliers. Thus, the Lagrangian has to be minimised with respect to the primal variables \mathbf{w} and b and maximized with respect to the dual variables $\alpha_i > 0$. The solution $(\mathbf{w}^*, b^*, \boldsymbol{\alpha}^*)$ should satisfy the K uhn-Tucker conditions

$$\frac{\partial \mathcal{L}(\mathbf{w}^*, b^*, \boldsymbol{\alpha}^*)}{\partial b} = 0 \quad (2.25a)$$

$$\text{and } \frac{\partial \mathcal{L}(\mathbf{w}^*, b^*, \boldsymbol{\alpha}^*)}{\partial \mathbf{w}} = 0. \quad (2.25b)$$

Using the conditions in (2.25), the primal variables \mathbf{w} and b can be expressed in terms of the dual variables $\boldsymbol{\alpha}$

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0 \quad (2.26)$$

$$\text{and } \mathbf{w} = \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i. \quad (2.27)$$

When substituted into (2.24), the resulting Wolf dual of the optimisation problem is given by

$$\text{maximise } \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,i}^{\ell} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle, \quad (2.28a)$$

$$\text{subject to } \sum_{i=1}^{\ell} \alpha_j y_i = 0, \quad (2.28b)$$

$$\alpha_i \geq 0, \quad i = 1, \dots, \ell. \quad (2.28c)$$

Let $\boldsymbol{\alpha}^* = (\alpha_1^*, \dots, \alpha_{\ell}^*)$ be the solution of the QP problem above. The vectors for which the corresponding Lagrange multipliers are positive are called *support vectors*. The values of the Lagrange multipliers assign weights to the corresponding vectors. These vectors and their weights are then used to define the decision rule or model. Therefore the learning machine in (2.28) is called the *support vector machine*.

The decision rule for the classification problem is then expressed in terms of the set of support vectors SV as

$$f(\hat{\mathbf{x}}) = \text{sgn} \left(\sum_{i \in SV} y_i \alpha_i^* \langle \mathbf{x}_i, \hat{\mathbf{x}} \rangle - b^* \right), \quad (2.29)$$

where b^* is the constant threshold or bias determined by

$$b^* = \frac{1}{2} [\langle \mathbf{w}^*, \mathbf{x}_1^* \rangle + \langle \mathbf{w}^*, \mathbf{x}_{-1}^* \rangle], \quad (2.30)$$

with \mathbf{x}_1^* any support vector belonging the class with output data 1, and \mathbf{x}_{-1}^* any support vector belonging to the class with output data -1 .

In Figure 2.7 the support vectors are the vectors on the margin, indicated with black markers. The support vectors are the vectors that lie on the margin and they typically represent the input data that are the most difficult to classify.

The type of optimal separating hyperplane described here, is a special case of a class of separating hyperplanes called the Δ -margin separating hyperplanes. It is proven that the set of Δ -margin separating hyperplanes has a bounded VC-dimension and that the classification error is bounded [108].

Non-separable case

Recall that it was assumed that the learning data set was indeed a separable set. Suppose that the data set is not separable. There are no hyperplanes that can separate the data set without error and no maximal margin can be constructed as seen in Figure 2.8. The slack variables ξ_i are introduced to penalise for those vectors lying within the margin. The value of a slack variable is the distance of the corresponding vector to the margin.

Since no hyperplane can be constructed without error, the optimisation problem is one minimizing the training errors:

$$\text{minimise } F_\sigma(\xi) = \sum_{i=1}^{\ell} \xi_i^\sigma, \quad (2.31a)$$

$$\text{subject to } y_i [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] \geq 1 - \xi_i, \quad i = 1, \dots, \ell, \quad (2.31b)$$

and using the structure

$$S_k = \left\{ \langle \mathbf{w}, \mathbf{x} \rangle + b : \|\mathbf{w}\|^2 \leq c_k \right\}. \quad (2.31c)$$

The minimisation of (2.31) with $\sigma = 1$ constructs a *soft margin* hyperplane. The optimisation problem stated in (2.31) is convex and can be expressed as a quadratic optimisation problem [106]. In the QP formulation the empirical risk (the errors) and the norm of the weights are minimised simultaneously, i.e.

$$\text{minimise } \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{\ell} \xi_i, \quad (2.32a)$$

$$\text{subject to } y_i [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] \geq 1 - \xi_i, \quad i = 1, \dots, \ell, \quad (2.32b)$$

$$\xi_i \geq 0, \quad i = 1, \dots, \ell. \quad (2.32c)$$

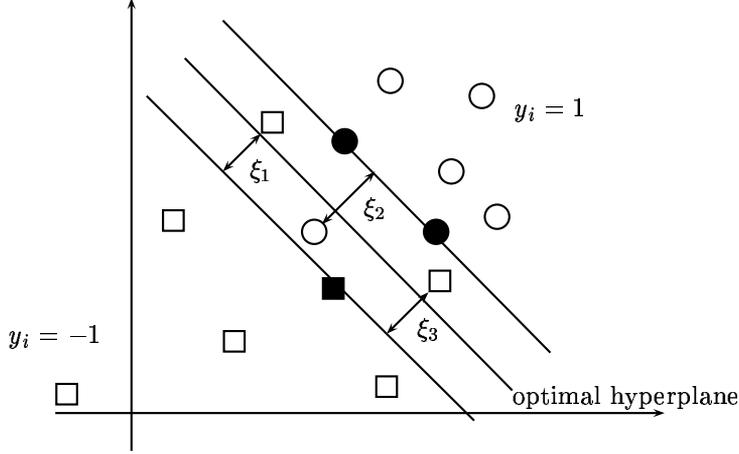


Figure 2.8: The Optimal Hyperplane for the non-separable case.

The parameter C affects the trade-off between complexity and the proportion of non-separable samples and is often referred to as the regularisation parameter.

Again by introducing Lagrange multipliers, finding the saddle point of the Lagrangian and writing the Wolf Dual problem, the QP problem to solve becomes,

$$\text{maximise } \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle, \quad (2.33a)$$

$$\text{subject to } \sum_{i=1}^{\ell} \alpha_i y_i = 0, \quad (2.33b)$$

$$0 \leq \alpha_i \leq C, \quad i = 1, \dots, \ell. \quad (2.33c)$$

Let SV be the set of the support vectors, then the resulting decision rule for the non-separable classification problem is expressed in terms of the support vectors as

$$f(\hat{\mathbf{x}}) = \text{sgn} \left(\sum_{i \in SV} y_i \alpha_i^* \langle \mathbf{x}_i, \hat{\mathbf{x}} \rangle - b^* \right), \quad (2.34)$$

where b^* is the constant threshold or bias determined by

$$b^* = \frac{1}{2} [\langle \mathbf{w}^*, \mathbf{x}_1^* \rangle + \langle \mathbf{w}^*, \mathbf{x}_{-1}^* \rangle], \quad (2.35)$$

with \mathbf{x}_1^* any support vector belonging to the class with output data 1 and \mathbf{x}_{-1}^* any support vector belonging to the class with output data -1 .

Finally, for SVM for classification to take place, the linear inner product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ in (2.33) (and (2.28)) is replaced by a (nonlinear) kernel function

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle.$$

The resulting decision rule for then becomes

$$f(\hat{\mathbf{x}}) = \text{sgn} \left(\sum_{i \in SV} y_i \alpha_i^* K(\mathbf{x}_i, \hat{\mathbf{x}}) - b^* \right). \quad (2.36)$$

The optimisation problems in (2.28) and (2.33) are solved by using standard optimisation routines. These routines typically define the optimisation problem in terms of the Hessian matrix and separate matrices for the equality and inequality constraints, as shown below.

Define $\mathbf{1}_n$ as an $n \times 1$ vector of ones and $\mathbf{0}_n$ an $n \times 1$ vector of zeros. Let \mathbf{H} be an $\ell \times \ell$ matrix such that

$$H_{i,j} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j),$$

then the SVM for classification in matrix notation is to minimise

$$\frac{1}{2} \boldsymbol{\alpha}^T \mathbf{H} \boldsymbol{\alpha} - \mathbf{1}_\ell^T \boldsymbol{\alpha} \quad (2.37)$$

subject to the equality constraint

$$\mathbf{y}^T \boldsymbol{\alpha} = 0 \quad (2.38)$$

and the inequality constraints

$$\mathbf{0}_\ell \leq \boldsymbol{\alpha} \leq C \mathbf{1}_\ell. \quad (2.39)$$

The algorithmic pseudo code for the SVM for classification can be found in Algorithm A.

Algorithm A. SVM for Classification

- Step 1:** Select the learning data $(\mathbf{x}_i, y_i), i = 1, \dots, \ell$, kernel function K and C .
- Step 2:** Construct the matrices for the QP problem.
- Step 3:** Solve (2.37) subject to (2.38) and (2.39) for $\boldsymbol{\alpha}$.
- Step 4:** Calculate the bias b using (2.35).
- Step 5:** Calculate $\|\mathbf{w}\|^2 = \boldsymbol{\alpha}^T \mathbf{H} \boldsymbol{\alpha}$.
- Step 6:** Calculate size of margin $\gamma = \frac{2}{\|\mathbf{w}\|}$
- Step 7:** Identify support vectors as $\{\mathbf{x}_i | \alpha_i > 0, i = 1, \dots, \ell\}$.
- Step 8:** Construct the index set SV of the support vectors.

Step 9: Construct the SVM Classification model using the determined parameters

$$f_{\alpha,b}(\hat{\mathbf{x}}) = \text{sgn} \left(\sum_{i \in SV} y_i \alpha_i K(\mathbf{x}_i, \hat{\mathbf{x}}) + b \right).$$

2.3.2 Regression case

The SVM method can also be used to solve regression estimation problems. Recall that the SVM method implements the SRM principle by keeping the empirical risk fixed or at a certain level whilst minimising the VC-dimension through the norm of the weights. However, in certain cases (like the non-separable classification) it is not possible to keep the empirical risk fixed. The errors are then penalised by adding slack variables and the optimisation problem then minimises the norm of the weights and the empirical error simultaneously [106],[11].

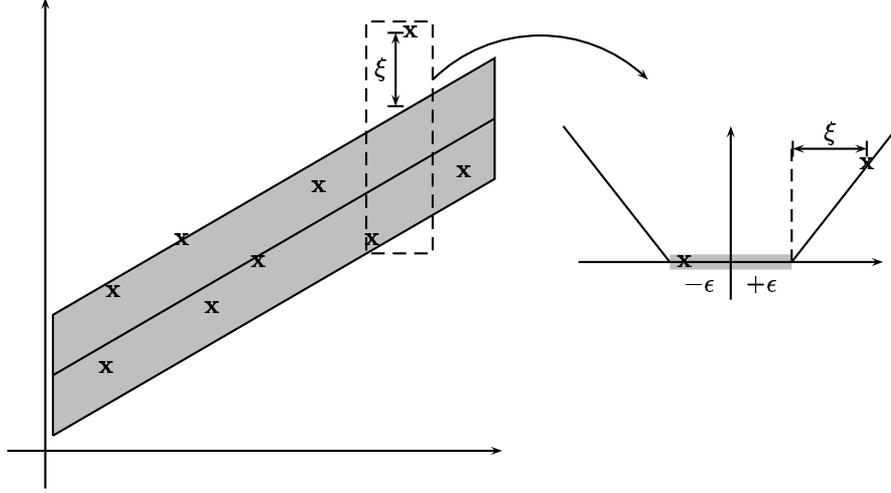
To measure the error of approximation, a loss function is used. Recall that different loss functions result in different models. The classic squared loss (L_2 norm) is used in least squares methods, and the absolute loss (L_1 norm) is used in the least modulus methods [11],[88]. Robust regression uses Huber's loss function which is related to the absolute loss function [27]. The SVM for regression, however uses the ϵ -insensitive loss function [106]

$$L^\epsilon(\mathbf{x}, y, f(\mathbf{x})) = |y - f(\mathbf{x})|_\epsilon = \max(0, |y - f(\mathbf{x})| - \epsilon), \quad (2.40)$$

where f is a real-valued function as defined in (2.17) and the learning data

$$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell), \mathbf{x}_i \in \mathbb{R}^n, y_i \in \mathbb{R}. \quad (2.41)$$

Through the introduction of the ϵ -insensitivity, the value of the empirical risk during training is kept fixed at an acceptable level, as is required by the second approach of implementing the SRM principle. By minimising the norm of the weights the Support Vector Machine for the regression can be formulated. However, as in the non-separable classification case, not all errors can be fixed absolutely to ϵ . Therefore, during the optimisation any error beyond the ϵ -level will be penalised as seen in Figure 2.9. There is also another reason for using the ϵ -insensitive loss function: *sparse representation* in terms for support vectors. This issue is readdressed later in the chapter.

Figure 2.9: The ϵ -insensitive zone.**Implicit complexity control**

The SVM for regression has to find \mathbf{w} and b that

$$\text{minimise } F(\xi, \xi^*) = \frac{1}{\ell} \left(\sum_{i=1}^{\ell} \xi_i + \xi_i^* \right), \quad (2.42a)$$

subject to

$$y_i - [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] \leq \epsilon + \xi_i, \quad i = 1, \dots, \ell, \quad (2.42b)$$

$$[\langle \mathbf{w}, \mathbf{x}_i \rangle + b] - y_i \leq \epsilon + \xi_i^*, \quad i = 1, \dots, \ell, \quad (2.42c)$$

$$\xi_i, \xi_i^* \geq 0, \quad i = 1, \dots, \ell, \quad (2.42d)$$

and using the structure

$$S_k = \left\{ \langle \mathbf{w}, \mathbf{x} \rangle + b : \|\mathbf{w}\|^2 \leq c_k \right\}. \quad (2.42e)$$

The optimisation problem in (2.42) is convex and can be written as the following QP problem [106],[11].

$$\text{minimise } \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{\ell} \sum_{i=1}^{\ell} (\xi_i + \xi_i^*), \quad (2.43a)$$

$$\text{subject to } (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - y_i \leq \epsilon + \xi_i, \quad i = 1, \dots, \ell, \quad (2.43b)$$

$$y_i - (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \leq \epsilon + \xi_i^*, \quad i = 1, \dots, \ell, \quad (2.43c)$$

$$\xi_i, \xi_i^* \geq 0, \quad i = 1, \dots, \ell. \quad (2.43d)$$

Again through introducing Lagrange multipliers, the Lagrange functional can be written,

$$\begin{aligned}
\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}, \boldsymbol{\alpha}^*) &= \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{\ell} \sum_{i=1}^{\ell} (\xi_i + \xi_i^*) \\
&\quad + \sum_{i=1}^{\ell} \alpha_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b - y_i - \epsilon - \xi_i) \\
&\quad + \sum_{i=1}^{\ell} \alpha_i^* (y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b - \epsilon - \xi_i^*) \\
&\quad - \sum_{i=1}^{\ell} (\eta_i \xi_i + \eta_i^* \xi_i^*). \quad (2.44)
\end{aligned}$$

Finding the saddle point of the Lagrangian with respect to the primal variables \mathbf{w} , b , $\boldsymbol{\xi}$ and $\boldsymbol{\xi}^*$, leads to

$$\mathbf{w} = \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) \mathbf{x}_i, \quad (2.45)$$

$$\sum_{i=1}^{\ell} \alpha_i - \alpha_i^* = 0, \quad (2.46)$$

$$\alpha_i = \frac{C}{\ell} - \eta_i, \quad i = 1, \dots, \ell, \quad (2.47)$$

$$\text{and } \alpha_i^* = \frac{C}{\ell} - \eta_i^*, \quad i = 1, \dots, \ell. \quad (2.48)$$

Making use of the fact that $\alpha_i \geq 0$ and $\eta_i \geq 0$, (2.47) can be written as

$$0 \leq \alpha_i \leq \frac{C}{\ell}, \quad i = 1, \dots, \ell. \quad (2.49)$$

Similarly for $\alpha_i^* \geq 0$ and $\eta_i^* \geq 0$, (2.48) is written as

$$0 \leq \alpha_i^* \leq \frac{C}{\ell}, \quad i = 1, \dots, \ell. \quad (2.50)$$

Substituting the equations in (2.45)-(2.48) into (2.44), we obtain the Wolf Dual pro-

blem

$$\begin{aligned} \text{maximise} \quad & -\frac{1}{2} \sum_{i,j=1}^{\ell} (\alpha_i^* - \alpha_i) (\alpha_j^* - \alpha_j) \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ & + \sum_{i=1}^{\ell} y_i (\alpha_i^* - \alpha_i) - \epsilon \sum_{i=1}^{\ell} (\alpha_i^* + \alpha_i), \end{aligned} \quad (2.51a)$$

$$\text{subject to} \quad \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) = 0, \quad (2.51b)$$

$$0 \leq \alpha_i, \alpha_i^* \leq \frac{C}{\ell}, \quad i = 1, \dots, \ell. \quad (2.51c)$$

Finally, for support vector regression to take place, the linear inner product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ in (2.51) has to be replaced by a (nonlinear) kernel function $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$. The best approximating function or model has then the form

$$f_{(\alpha^* - \alpha), b}(\hat{\mathbf{x}}) = \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) K(\mathbf{x}_i, \hat{\mathbf{x}}) + b, \quad (2.52)$$

where b is the bias determined by

$$b = \frac{1}{|S|} \sum_{i \in S} \left(y_i - \sum_{j=1}^{\ell} (\alpha_j^* - \alpha_j) K(\mathbf{x}_j, \mathbf{x}_i) - \epsilon \operatorname{sgn}(\alpha_i^* - \alpha_i) \right), \quad (2.53)$$

where S contains the indexes of the Lagrange multipliers for which $0 < |\alpha_i^* - \alpha_i| < C/\ell$ holds.

In SVM for regression the parameter ϵ controls the complexity implicitly, whereas the parameter C is used to control the trade-off between the complexity and the error terms. An estimation method for determining a suitable value for C , is derived and discussed in Chapter 4.

The optimisation problem in (2.51) is solved by using standard optimisation routines that typically define the QP problem in terms of the Hessian matrix and separate matrices for the equality and inequality constraint, as shown below.

The Hessian matrix \mathbf{H} is now an $2\ell \times 2\ell$ matrix such that

$$H = \begin{pmatrix} K(\mathbf{x}, \mathbf{x}) & -K(\mathbf{x}, \mathbf{x}) \\ -K(\mathbf{x}, \mathbf{x}) & K(\mathbf{x}, \mathbf{x}) \end{pmatrix}.$$

For $\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\alpha}^* \\ \boldsymbol{\alpha} \end{bmatrix}$ the SVM for regression in matrix notation is then to minimise

$$\frac{1}{2} \boldsymbol{\beta}^T \mathbf{H} \boldsymbol{\beta} + \begin{bmatrix} \epsilon \mathbf{1}_{\ell} - \mathbf{y} \\ \epsilon \mathbf{1}_{\ell} + \mathbf{y} \end{bmatrix}^T \boldsymbol{\beta} \quad (2.55)$$

subject to the equality constraint

$$\begin{bmatrix} \mathbf{1}_{\ell} \\ -\mathbf{1}_{\ell} \end{bmatrix}^T \boldsymbol{\beta} = 0 \quad (2.56)$$

and the inequality constraints

$$\mathbf{0}_{2\ell} \leq \boldsymbol{\beta} \leq \frac{C}{\ell} \mathbf{1}_{2\ell}. \quad (2.57)$$

The algorithmic pseudo code of the ϵ -SVM for regression can be found in Algorithm B.

Algorithm B. ϵ -SVM for regression

- Step 1:** Select the learning data $(\mathbf{x}_i, y_i), i = 1, \dots, \ell$, kernel function K, C and *epsilon*.
- Step 2:** Construct matrices for the QP problem.
- Step 3:** Solve (2.55) subject to (2.56) and (2.57) for $\boldsymbol{\beta}$.
- Step 4:** Calculate $\boldsymbol{\alpha}^* = [\beta_1, \dots, \beta_\ell]$ and $\boldsymbol{\alpha} = [\beta_{\ell+1}, \dots, \beta_{2\ell}]$.
- Step 5:** Calculate the bias b using (2.53).
- Step 6:** Calculate $\|\mathbf{w}\|^2 = (\boldsymbol{\alpha}^* - \boldsymbol{\alpha})^T H (\boldsymbol{\alpha}^* - \boldsymbol{\alpha})$.
- Step 7:** Identify support vectors as $\{\mathbf{x}_i \mid (\boldsymbol{\alpha}_i^* - \boldsymbol{\alpha}_i) \neq 0, i = 1, \dots, \ell\}$.
- Step 8:** Construct the index set SV of the support vectors.
- Step 9:** Set $\boldsymbol{\omega} = (\boldsymbol{\alpha}^* - \boldsymbol{\alpha})$.
- Step 10:** Construct the SVM regression model using the determined parameters

$$f_{\boldsymbol{\omega}, b}(\hat{\mathbf{x}}) = \sum_{i \in SV} \omega K(\mathbf{x}_i, \hat{\mathbf{x}}) + b.$$

Explicit complexity control

One of the main drawbacks of the ϵ -insensitive SVM for regression is that one has to choose the value of ϵ *a priori*. The value of ϵ defines the size of the insensitive zone which indicates what the desired accuracy of the approximation is. Often one does not want to be committed to a fixed value of ϵ , but to something which is as accurate as possible.

This is achieved by introducing a parameter ν which scales the value of ϵ [89],[78]. Consider the loss functional

$$\begin{aligned} L^\epsilon &= \frac{C}{\ell} \sum_{i=1}^{\ell} |y_i - f(\mathbf{x}_i)|_\epsilon \\ &= \frac{C}{\ell} \sum_{i=1}^{\ell} \max(0, (|y_i - f(\mathbf{x}_i)| - \epsilon)) \\ &= \frac{C}{\ell} \sum_{i=1}^{\ell} \begin{cases} (y_i - f(\mathbf{x}_i) - \epsilon) & : y_i - f(\mathbf{x}_i) \geq \epsilon \\ (f(\mathbf{x}_i) - y_i - \epsilon) & : y_i - f(\mathbf{x}_i) \leq -\epsilon \\ 0 & : \text{otherwise} \end{cases} \end{aligned}$$

If ϵ is unknown, but we know that the ratio ν scales the value of ϵ ,

$$\begin{aligned} L^{\epsilon, \nu} &= \frac{C}{\ell} \sum_{i=1}^{\ell} \begin{cases} \xi_i + \nu\epsilon & : y_i - f(\mathbf{x}_i) \geq \epsilon \\ \xi_i^* + \nu\epsilon & : y_i - f(\mathbf{x}_i) \leq -\epsilon \\ \nu\epsilon & : \text{otherwise} \end{cases} \\ &= \frac{1}{\ell} \sum_{i=1}^{\ell} (\xi_i + \xi_i^* + \ell\nu\epsilon) \\ &= \nu\epsilon + \frac{1}{\ell} \sum_{i=1}^{\ell} (\xi_i + \xi_i^*) \end{aligned}$$

Replacing the original loss functional with the one above, the so-called ν -SVM for regression is obtained. The QP problem for the ν -SVM is

$$\text{minimise} \quad \frac{1}{2} \|\mathbf{w}\|^2 + C \left(\nu\epsilon + \frac{1}{\ell} \sum_{i=1}^{\ell} (\xi_i + \xi_i^*) \right), \quad (2.58a)$$

$$\text{subject to} \quad (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - y_i \leq \epsilon + \xi_i, \quad i = 1, \dots, \ell, \quad (2.58b)$$

$$y_i - (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \leq \epsilon + \xi_i^*, \quad i = 1, \dots, \ell, \quad (2.58c)$$

$$\xi_i \geq 0, \quad \xi_i^* \geq 0, \quad \epsilon \geq 0, \quad i = 1, \dots, \ell. \quad (2.58d)$$

The resulting Wolf Dual problem then can be written as

$$\text{maximise} \quad -\frac{1}{2} \sum_{i,j=1}^{\ell} (\alpha_i^* - \alpha_i) (\alpha_j^* - \alpha_j) \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \sum_{i=1}^{\ell} y_i (\alpha_i^* - \alpha_i), \quad (2.59a)$$

$$\text{subject to} \quad \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) = 0, \quad (2.59b)$$

$$\sum_{i=1}^{\ell} (\alpha_i^* + \alpha_i) \leq C \cdot \nu, \quad (2.59c)$$

$$0 \leq \alpha_i^{(*)} \leq \frac{C}{\ell}, \quad i = 1, \dots, \ell. \quad (2.59d)$$

The parameter ν replaces the ϵ and is the ratio that varies between 0 and 1. Using ν makes it much easier to find the best value for ν than for an ϵ that is only bounded from below by 0.

For support vector regression to take place, the linear inner product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ in (2.59) has to be replaced by a (nonlinear) kernel function $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$. The best approximating function or model has the same form as for the ϵ -SVM,

$$f_{(\alpha^* - \alpha), b}(\hat{\mathbf{x}}) = \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) K(\mathbf{x}_i, \hat{\mathbf{x}}) + b, \quad (2.60)$$

where b is the bias determined by (2.53).

We now write the optimisation problem in (2.59) in matrix notation. Minimise

$$\frac{1}{2} \boldsymbol{\beta}^T \mathbf{H} \boldsymbol{\beta} + \begin{bmatrix} -\mathbf{y} \\ \mathbf{y} \end{bmatrix}^T \boldsymbol{\beta} \quad (2.61)$$

subject to the equality constraint

$$\begin{bmatrix} \mathbf{1}_{\ell} \\ -\mathbf{1}_{\ell} \end{bmatrix}^T \boldsymbol{\beta} = 0 \quad (2.62)$$

and the inequality constraints

$$\begin{bmatrix} \mathbf{0}_{2\ell} \\ 0 \end{bmatrix} \leq \begin{bmatrix} \boldsymbol{\beta} \\ (\mathbf{1}_{2\ell})^T \boldsymbol{\beta} \end{bmatrix} \leq \begin{bmatrix} (C/\ell) \mathbf{1}_{2\ell} \\ C \cdot \nu \end{bmatrix}. \quad (2.63)$$

The algorithmic pseudo code of the ν -SVM for regression can be found in Algorithm C.

Algorithm C. ν -SVM for Regression

- Step 1:** Select the learning data $(\mathbf{x}_i, y_i), i = 1, \dots, \ell$, kernel function K , C and ν .
 - Step 2:** Construct the matrices for the QP problem.
 - Step 3:** Solve (2.61) subject to (2.62) and (2.63) for $\boldsymbol{\beta}$.
 - Step 4:** Steps 4-10 of Algorithm B.
-

Recall that the reason for using the ϵ -insensitive is two-fold. We already explained the one reason, namely that the second approach of implementation of the SRM principle requires the error to be fixed at a certain level. That is achieved by using the ϵ -insensitive zone. The second reason, namely that it provides sparse representation, we did not explain yet. Now consider the SVM regression model given by

$$f_{\omega, b}(\hat{\mathbf{x}}) = \sum_{i \in SV} \omega_i K(\mathbf{x}_i, \hat{\mathbf{x}}) + b, \quad (2.64)$$

where ω and b are the weights and bias determined by the SVM using the kernel function K and input data \mathbf{x} , and SV is the index set of the support vectors. If $|SV| < \ell$, it is clear that the number of terms in a SVM model is fewer than the number of terms that would have been used in a model built on the basis of one of the traditional loss functions⁵.

2.4 Choice of complexity parameters

The complexity can be controlled implicitly or explicitly. The implicit method controls the number of support vectors by controlling the acceptable error level through ϵ . Using ν , the ratio of support vectors can be controlled explicitly.

Whether the implicit or explicit complexity control is used, the complexity parameter has to be set *a priori*. This is only necessary in the case of regression. By definition, the complexity is automatically minimised in support vector classification [106].

The choice of the complexity parameter depends on various factors. In the following paragraphs the effect of changing the parameter, what influences the parameter and how to determine the parameter will be discussed.

2.4.1 Size of the insensitive zone (ϵ)

Through the value ϵ the size of an insensitive zone around the hyperplane is controlled. We assume that any approximation error smaller than ϵ is due to noise and accept it [88],[91]. Furthermore, the value of ϵ should reflect the noise variance in the learning data[11]. Therefore one can see this insensitive zone as a sort of acceptable noise level. During the learning phase, these approximation errors will be ignored and therefore the method is said to be insensitive to errors inside this zone.

All approximation errors outside this zone are not due to noise and will be taken into account during optimisation. The vectors (data points) that have approximation errors outside the insensitive zone, are the most difficult to predict. These vectors contain most of the information and are important. Therefore, the support vectors in the regression case are those vectors that lie outside the insensitive zone. By decreasing the value of ϵ , the number of support vectors is increased since more and more vectors lie outside the insensitive zone. Throughout the thesis the following example (or variations of it) will be used to illustrate ideas and results.

Example 1. (Sin function with added noise.)

Step 1: Generate 100 data points x_i using a uniform distribution over $[0, 1]$.
Step 2: Generate 100 random noise values η_i using $N(0, 0.07)$.
Step 3: The output values y_i were determined as $y_i = \sin(2\pi x_i)^2 + \eta_i$.

In Figure 2.10 the effect of ϵ is shown using Example 1 and the parameters $C = 1000$ and a radial basis function (RBF) kernel with kernel parameter $\sigma = 0.25$. One can see that for increasing values of ϵ , the size of the ϵ -insensitive zone increases such

⁵Here $|A|$ is the cardinality of the set A .

that fewer data points lie outside the zone resulting in fewer support vectors chosen. Furthermore, one can see that for this example it is not necessary to use a large number of support vectors in order to achieve good predictability. Since the number of support vectors, which determines the complexity of the resulting model, is not known beforehand, but is controlled through ϵ , the complexity is said to be controlled implicitly.

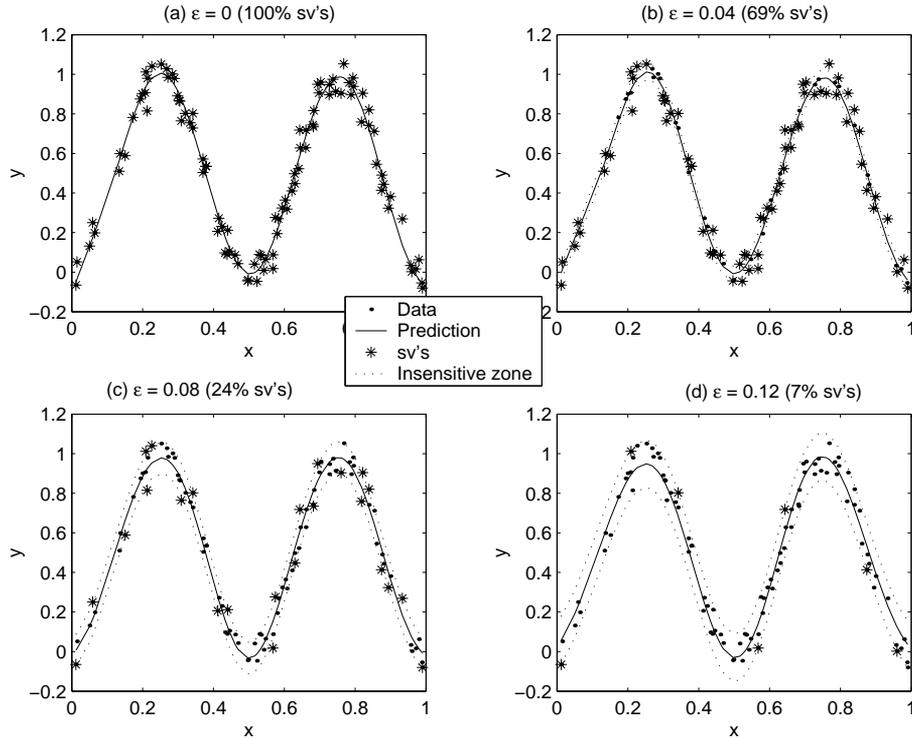


Figure 2.10: Increasing values of ϵ lead to a reduction in the number of support vectors used by the model.

The optimal value of ϵ can be found by inspecting various levels of complexity and selecting the parameter for which the resulting model has the best performance with respect to generalisation error as well as approximation error [39].

The algorithmic pseudo code for the optimisation of ϵ is given in Algorithm D.

Algorithm D. Optimisation of ϵ

-
- Step 1:** Select the learning data $(\mathbf{x}_i, y_i), i = 1, \dots, \ell$, kernel function K , C and iteration parameter $p = [\epsilon_start, \epsilon_end, it_num]$.
- Step 2:** Construct a vector ϵ consisting of it_num values linearly distributed from ϵ_start to ϵ_end .

- Step 3:** For $k = 1$ to it_num
 run Algorithm B using ε_k
 set $\mathbf{w}_k = \mathbf{w}$
 set $b_k = b$
 set $SV_k = \{SV\}$
 construct the model $f_k(\hat{\mathbf{x}}) = \mathbf{w}_k^T K(\mathbf{x}, \hat{\mathbf{x}}) + b_k$
 Calculate the predictions $\mathbf{py}_k = f_k(\mathbf{x})$.
 end.
- Step 4:** Find the optimal model f^* using Vapnik's measure (See Chapter 9).
- Step 5:** Return optimal $\epsilon = \varepsilon^*$.
-

2.4.2 Ratio of support vectors (ν)

Through the ratio ν the desired ratio of support vectors to be used is chosen *a priori*. The size of the ϵ -insensitive zone is determined automatically [73],[88],[10],[89].

Furthermore, if $\nu > 1$, then ϵ must be 0, since it does not pay to increase ϵ . This can be seen from (2.58a) where using a value $\nu > 1$ makes the slack variables become less important. Another way to see this, is to note that in (2.58c) a $\nu \geq 1$ implies (2.58d) since, if all $\alpha_i^{(*)}$ hit the upper boundary C/ℓ and we take into account that $\alpha_i \alpha_i^* = 0$ for all i , the sum over $i = 1, \dots, \ell$ of the α 's has a supremum of $\ell \cdot (C/\ell) = C$. Thus, the constraint in (2.58d) is redundant for all values of $\nu \geq 1$. In fact all values of $\nu \geq 1$ are equivalent. Therefore, the values of ν are between 0 and 1.

For $\nu < 1$, the value of ϵ are usually larger than zero. In noise free data, which can be approximated without error with a low complexity model, ϵ can still be found to be zero. Of course, the case of $\epsilon = 0$ is not very interesting since we lose the sparse representation character of SVMs and this is equivalent to the plain L_1 -loss regression. For $\epsilon > 0$, the following statements hold (for the proof the reader is referred to [73],[88]):

1. ν is an upper bound on the fraction of errors.
2. ν is a lower bound on the fraction of support vectors.
3. Suppose the data in S , were generated i.i.d. from a distribution $P(\mathbf{x}, y) = P(\mathbf{x})P(y|\mathbf{x})$ with $P(y|\mathbf{x})$ continuous. With probability 1, asymptotically, ν equals both the fraction of support vectors and the fraction of errors.

In Figure 2.11 the effect of ν is shown using Example 1 and parameter set to $C = 1000$ and a RBF kernel with kernel parameter $\sigma = 0.25$. One can see that for increasing values of ν , the number of support vectors decreases as the size of the ϵ -insensitive zone increases. Since the value of ν gives an indication of the ratio of support vectors, the complexity is said to be controlled explicitly. It is difficult to say what a good choice for ν is. For a data set with moderate noise level and few outliers, a reasonable

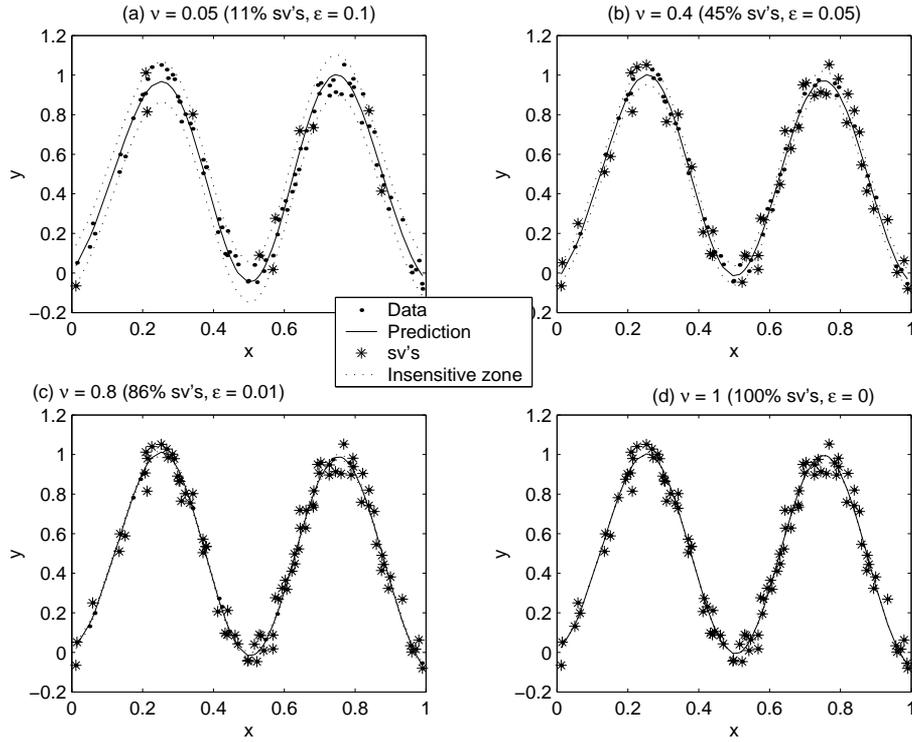


Figure 2.11: Increasing values of ν lead to decreasing values of ϵ . Consequently the number of support vectors used by the model is increased.

value lies in the region of 0.5 [55]. To find the optimal value of ν , various levels of complexity have to be inspected and the parameter for which the resulting model has the best performance with respect to generalisation error as well as approximation error is selected.

The algorithmic pseudo code for the optimisation of ν is very similar to Algorithm D where ν is used in stead of ϵ . It only differs in Step 3 where Algorithm B is replaced by Algorithm C.

2.5 Conclusion

Let us state the original design requirement again:

The inferential sensor should be able to use the optimal complexity for the given data set as well as control it.

In order to achieve this design requirement, some important results from SLT were discussed. A simple way to summarise these results is as follows, where the italic

text corresponds to the structure below. The classical ERM $R_{\text{emp}}(\alpha)$ finds the *best approximating function*. Learning machines that implement the ERM therefore have control over the *approximation error*. However, due the consistency of the ERM principle, the learning machines can only be applied to *large sample sizes*. In order to learn from a *small sample size* data set, the learning machine is required to have *control over the complexity* of the model. The control then enables the learning machine to select the model with the *best structure/capacity* for the given sample size. The term $R_{\text{emp}}(\alpha)$ is concerned with the *learning ability* of the learning machine and is used to assess the *quality of the approximation*. The term $\Phi(\ell/h)$ is concerned with the *complexity of the model* constructed by the learning machine and is used to assess the *quality of generalisation*. Therefore, the minimisation of the combined terms is the SRM principle. These important aspects are used in constructing the SRM principle, which can be summarised as follows.

$$\begin{array}{ccc}
 \min R(\alpha) = & R_{\text{emp}}(\alpha) & + & \Phi(\ell/h) \\
 & \Downarrow & & \Uparrow \\
 & \text{Best approximating function} & & \text{Best structure/capacity} \\
 & \Downarrow & & \Uparrow \\
 & \text{Control approximation error} & & \text{Control complexity}(h) \\
 & \Downarrow & & \Uparrow \\
 & \underbrace{\text{Large sample sizes}} & & \underbrace{\text{Small sample sizes}(\ell)} \\
 & \text{Learning Ability} & & \text{Complexity of Model} \\
 & \Downarrow & & \Uparrow \\
 & \text{Quality of Approximation} & \iff & \text{Quality of Generalisation}
 \end{array}$$

One implementation of the SRM principle is SVMs, which gives the direct control over the complexity of the resulting model. Therefore, in the context of adaptive inferential sensor development, we conclude that the SVM method is preferable to other methods since it is the only learning machine that has the desirable characteristics required. These characteristics being the learning method's ability to minimise the complexity of the model, the ability to control the complexity of the model and that the learning method works well with small sample sized data sets. The fact that the SVM model is a sparse representation of the learning data is a bonus. Furthermore, the characteristics of support vectors can possibly be used to solve a number of other design requirements.

On the practical side, algorithmic pseudo codes for the SVM for classification, ϵ -SVM for regression and ν -SVM for regression were given. We further gave some practical insights into the influence of the complexity parameters ϵ and ν as well as how to select these parameters. Finally we discussed the optimisation of the complexity parameters and gave algorithmic pseudo codes as well.

Chapter 3

High-dimensional Data and Spaces

3.1 Introduction

The number of variables measured in a process plant has increased dramatically as computer software made obtaining and handling the thousands of measurements faster and easier. Furthermore, the industry needs to analyse their processes in order to understand exactly what influenced the quality of the production. The result is that many new hardware sensors were developed and implemented to measure various process variables. Currently it is not rare for analysts to receive data sets with hundreds of variables [39]. Therefore, one part of the second design requirement requires inferential sensors to be able to use high-dimensional data sets.

Since most of the problems encountered in real-life are nonlinear, linear learning machines map the input data into a higher dimensional feature space where the learning capacity of the linear learning machine is increased [106],[11]. However, as the number of dimensions grows, the dimensionality of the feature space can become computationally unmanageable. Also, with an increasing number of dimensions the number of data samples needed for a sufficiently high sampling density increases exponentially [11]. High sampling density is necessary to ensure that data covers the modelling space well. The phenomenon that the learning machine's computational and predictive performance can degrade as the number of input dimensions increases, is often referred to as the *curse of dimensionality* [10].

A common approach to find a way around this problem, is to limit the number of input dimensions to a smaller set of relevant dimensions. These relevant dimensions still need to consist the essential information contained by the original set. A well-known statistical method for dimensionality reduction is Principle Component Analysis (PCA) [52]. Other widely used methods in industry for dimensionality reduction are NNs [47] and GP [41]. However, even after such steps were taken industrial data sets may still have too many dimensions for classical learning machines. Therefore, the second part of the design requirement requires the inferential sensor to overcome

the curse of dimensionality.

In the first section of this chapter we briefly explain some important issues involving the curse of dimensionality. The second section deals with kernel functions. Their use in SVMs results in an *implicit* and not an *explicit* mapping of the learning data onto a higher dimensional space. Two different types of kernels, Radial Basis Function (RBF) kernels and polynomial kernels, are discussed in the third section. Finally, we explain in the fourth section how to choose of the type of kernel and determine the parameters used by the kernel function.

3.2 Curse of dimensionality

The goal of the learning machine is to estimate a function from a finite number of training data. In Chapter 2 we have seen that for good generalisation performance, the approximating function should be smooth enough, i.e. a small norm in coefficients. The constraints imposed on the smoothness of the function define the possible behaviour of the function in local neighbourhoods of the input space. It is obvious that the accuracy of the function estimation will depend on whether or not there are enough samples within the neighbourhood. In high-dimensional input spaces, it is difficult to collect enough samples for an adequate sampling density. The result is that the learning machine often does not have enough information available in order to learn and predict well.

Linear learning machines also need to map the input data into a higher dimensional feature space where the learning capacity of the learning machine is increased [106]. As the number of input dimensions increases, the dimensionality of the feature space can become immense and even infinite. The problem could then become computationally unmanageable[11].

The degradation in the learning machine's computational and predictive performances when the number of input dimensions increases, given a fixed number of samples, is called the curse of dimensionality [11], [10],[106].

Curse of dimensionality with respect to predictive performance

The degradation of the predictive performance of a learning machine is mainly due to the geometry of high-dimensional space. The following properties show how for example high-dimensional distributions contribute to the curse of dimensionality [11].

Property 1: Sampling density. For increasing dimensions, the number of data points needed to retain the same sampling density, increases exponentially. For example, suppose a sample of ℓ data points in \mathbb{R}^1 is considered as a dense sample. Then, in order to obtain the same sampling density in \mathbb{R}^n , the number of data points needed is ℓ^n .

Property 2: Large neighbourhoods. For increasing dimensions, a larger radius is needed to enclose the same fraction of samples. For example, consider data points taken from a n -dimensional uniform distribution over the unit hypercube. In \mathbb{R} , the hypercube is a line of length 1; in \mathbb{R}^2 it is a one-by-one square;

in \mathbb{R}^3 it is a cube with edge lengths equal to one; and so fourth. Suppose that another smaller hypercube exists within the unit hypercube such that it encloses a predefined fraction p of the data points in the unit hypercube. Let us denote this smaller hypercube with Q^n . The edge length of Q^n for increasing dimensions is determined by

$$e_n(p) = p^{\frac{1}{n}}.$$

Suppose it is required that Q^n encloses 10% of the data points of the unit hypercube. The edge length for Q^n in \mathbb{R} is then 0.1 and in \mathbb{R}^2 it is 0.32. In \mathbb{R}^3 the edge length of Q^3 is 0.46. This can be seen in Figure 3.1. Continuing in this fashion, enclosing 10% of the data points in the unit hypercube in \mathbb{R}^{10} requires that Q^{10} has an edge length of 0.8. Thus, for high-dimensional spaces a very large part of the input space needs to be sampled in order to obtain only a small fraction of the data points.

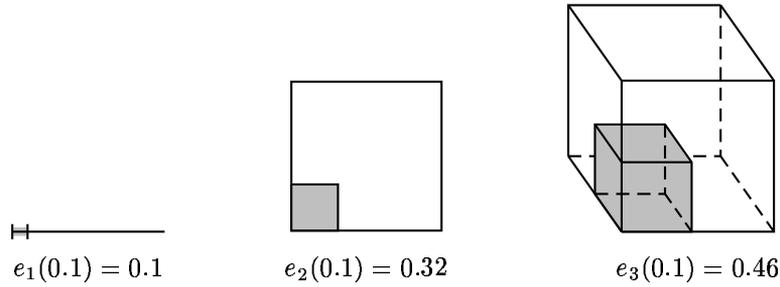


Figure 3.1: Increasing dimensions requires larger edge lengths for Q to enclose 10% of data points in the unit hypercube.

Property 3: *Distance to the edge of the space.* For high-dimensional spaces, almost every point is closer to the edge of the space than to another data point. Consider again the n -dimensional uniform distribution over the unit hypercube. The distance of any data point from the edge of the unit hypercube is at most 0.5. The maximum distance is achieved by the data point exactly at the centre of the unit hypercube, which is also the centre of the distribution. Now let us define for a sample of ℓ data points the expected distance, L_∞ , between data points as

$$D(n, \ell) = \frac{1}{2} \left(\frac{1}{\ell} \right)^{\frac{1}{n}}.$$

In \mathbb{R}^{10} , $D(10, 1000) \approx 0.5$ and $D(10, 10000) \approx 0.4$. These averages are very close to the maximum distance from any data point in the unit hypercube to the edge of the unit hypercube. This means that already in a ten-dimensional

space, most data points will be closer to the edge of the space than to another point.

Property 4: Distance from the centre of the space. In high-dimensional spaces, almost every data point looks like an outlier of the training data. Consider again the example of n -dimensional uniform distributions over unit hypercubes. Let us denote the unit hypercube in \mathbb{R}^n with Q_1^n . The number of corners for Q_1^n is 2^n . In Figure 3.1 one can see that Q_1^2 has 4 corners and Q_1^3 has 8 corners. The euclidian distance from the centre of the unit hypercube to each corner is

$$\sqrt{\sum_{i=1}^n \left(\frac{1}{2}\right)^2}.$$

Recall from Property (3) that the distance from the centre to the edge is 0.5. Now, using polar coordinates, the unit hypercubes can be visualised as seen in Figure 3.2. The number of corners is linearly distributed over 2π and the radius indicates the distance from the centre of Q_1^n to the corners and edges. As the number of dimensions increases, the distance from the centre to the corners grows longer relative to the size of the central spherical part of the distribution in Figure 3.2. For a data point at a corner (at the tip of a spike) the other data points would appear far away and concentrated in the sphere around the centre. Therefore, with respect to the other data points in the sphere (that is being near the centre of distribution), the data point appears to be an outlier.

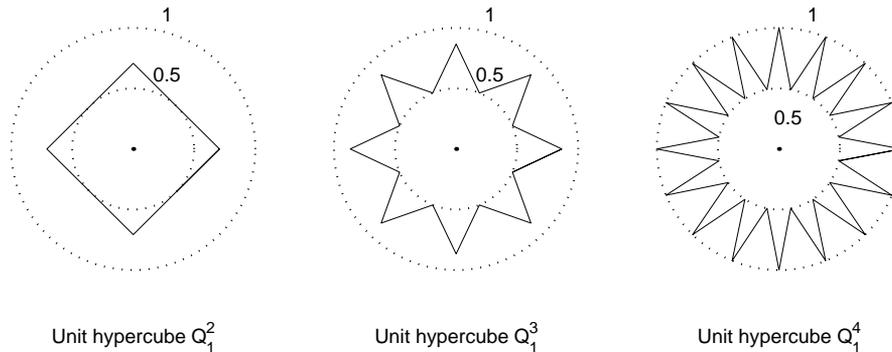


Figure 3.2: Unit hypercubes visualised using polar coordinates. Conceptually, a high-dimensional hypercube therefore looks like a star.

The properties 1 and 2 explain why it is difficult to make local estimates of high-dimensional data. The immense number of samples required for meaningful estimation in high-dimensional spaces, can seldom be obtained. The properties 3 and 4 indicate that in high-dimensional spaces predictions made by a learning machine will often result in an extrapolation. Since almost every point is closer to the edge than to another point, extrapolation becomes the rule rather than the exception.

Curse of dimensionality with respect to computational performance

The second part of the curse of dimensionality is the computational difficulties encountered by the learning machine in high-dimensional spaces. Recall that linear learning machines map the input data into a feature space through nonlinear transformations. For example, let $\phi_j(\mathbf{x}), j = 1, \dots, m$ denote a set of nonlinear transforms corresponding to polynomial terms of the components of \mathbf{x} up to a certain order (including the interaction terms). In \mathbb{R}^2 , the mapping of input data $\mathbf{x} = (x_1, x_2)$ using third-order polynomials, would result in a 16-dimensional feature space consisting of the following features,

$$\begin{array}{lll}
 \phi_1(x_1, x_2) = 1 & & \\
 \phi_2(x_1, x_2) = x_1 & \phi_3(x_1, x_2) = x_1^2 & \phi_4(x_1, x_2) = x_1^3 \\
 \phi_5(x_1, x_2) = x_2 & \phi_6(x_1, x_2) = x_2^2 & \phi_7(x_1, x_2) = x_2^3 \\
 \phi_8(x_1, x_2) = x_1 x_2 & \phi_9(x_1, x_2) = x_1^2 x_2 & \phi_{10}(x_1, x_2) = x_1^3 x_2 \\
 \phi_{11}(x_1, x_2) = x_1 x_2^2 & \phi_{12}(x_1, x_2) = x_1^2 x_2^2 & \phi_{13}(x_1, x_2) = x_1^3 x_2^2 \\
 \phi_{14}(x_1, x_2) = x_1 x_2^3 & \phi_{15}(x_1, x_2) = x_1^2 x_2^3 & \phi_{16}(x_1, x_2) = x_1^3 x_2^3
 \end{array}$$

In this higher dimensional feature space, the linear learning machine constructs an approximating function resulting in a third-order polynomial function in the input space. It is also clear from this example that for increasing dimensions in the input space, the dimensionality of the feature space can become very large. For learning machines that require the explicit calculation of the transformations in the feature space, for example $D(\mathbf{x}) = \sum_{j=1}^m w_j \phi_j(\mathbf{x})$, the task could become computationally unmanageable since the summation depends on the dimensionality of the feature space. Here the bias is omitted because the constant basis function $\phi_1(\mathbf{x}) = 1$ is included in the feature space.

Dimensionality reduction

The problems encountered with high-dimensional data sets lead to the whole field of dimensionality reduction. Dimensionality reduction is in principle an unsupervised learning problem [39] and is typically performed using PCA [52] and NNs[47]. In inferential sensor development, dimensionality reduction is seen as a preprocessing step to the main learning problem of function estimation [33],[41].

In practice NNs and GPs are often used to perform a sensitivity analysis on the various dimensions in the training data [41]. For each dimension a weight is determined that gives an indication of the relative importance of the dimension. The dimensions that have the highest importance are then selected as the reduced set of dimensions.

Often in the quest to reduce the number of dimensions, the learning method is left with learning data that lack a lot of information. The reduced set should account for a high percentage of the variability in the data. Therefore, performing dimensionality reduction is a balancing act of the number of dimensions against how

much information is lost. The more complex and nonlinear a problem becomes, the more difficult it is to reduce the number of dimensions to a manageable set.

However, sometimes even if dimensionality reduction was performed, the problem may still have too many dimensions for classical methods. Therefore, it is important that we look for methods that try to alleviate the effects of the curse of dimensionality.

3.3 Kernel functions

In many learning machines the learning data are mapped (nonlinearly) into a higher dimensional feature space where the computational power of the linear learning machine is increased [106]. The mathematics of the SVM given in Chapter 2 expresses these transformations as linear inner products of the input data. In order to make the mapping nonlinear, we make use of an *inner product* kernel which has the following form

$$K(\mathbf{x}, \hat{\mathbf{x}}) = \sum_{j=1}^m \phi_j(\mathbf{x})\phi_j(\hat{\mathbf{x}}), \quad (3.1)$$

for a given set of basis functions $\phi_j(\mathbf{x})$.

The inner product kernel in a Hilbert Space has the following general form

$$\langle z, \hat{z} \rangle = K(\mathbf{x}, \hat{\mathbf{x}}), \quad (3.2)$$

where \mathbf{z} is the image in the feature space of the vector \mathbf{x} in the input space. According to Hilbert-Schmidt theory, the convolution of the inner product kernel, $K(\mathbf{x}, \hat{\mathbf{x}})$, can be any symmetric function that satisfies Mercer's conditions [106]:

$$\int \int K(\mathbf{x}, \hat{\mathbf{x}})g(\mathbf{x})g(\hat{\mathbf{x}})d\mathbf{x}d\hat{\mathbf{x}} > 0, \quad \text{for all } g \neq 0, \quad \int g^2(\mathbf{x})d\mathbf{x} < \infty. \quad (3.3)$$

For finite dimensional input spaces, the conditions (3.3) states that $K(\mathbf{x}, \hat{\mathbf{x}})$ is an admissible kernel function if it produces a kernel matrix that is symmetric and positive semi-definite.

The SVM for regression method using a kernel function and the ϵ -insensitive loss function is then formulated as [106],

$$\begin{aligned} \text{maximise} \quad & \sum_{i=1}^{\ell} y_i (\alpha_i^* - \alpha_i) - \epsilon \sum_{i=1}^{\ell} (\alpha_i^* + \alpha_i) \\ & - \frac{1}{2} \sum_{i,j=1}^{\ell} (\alpha_i^* - \alpha_i) (\alpha_j^* - \alpha_j) K(\mathbf{x}_i, \mathbf{x}_j), \end{aligned} \quad (3.4a)$$

$$\text{subject to} \quad \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) = 0, \quad (3.4b)$$

$$0 \leq \alpha_i, \alpha_i^* \leq \frac{C}{\ell}, \quad i = 1, \dots, \ell. \quad (3.4c)$$

The approximating function is given in terms of the weights, the kernel and the bias, according

$$f(\hat{\mathbf{x}}) = \sum_{i \in SV} (\alpha_i^* - \alpha_i) K(\mathbf{x}_i, \hat{\mathbf{x}}) + b. \quad (3.5)$$

Using the kernel functions has two major advantages. In 1992 it was observed by Boser et al. that one does not need to consider the feature space in explicit form [4]. That is to determine all the features of \mathbf{x} and $\hat{\mathbf{x}}$ in the feature space and then determine the inner product $\langle \phi(\mathbf{x}), \phi(\hat{\mathbf{x}}) \rangle$. Through the use of the kernel function $K(\mathbf{x}, \hat{\mathbf{x}})$ the inner product in the feature space is determined directly as a function of the input data in the input space. Furthermore, the SVM model in (3.5) only uses the support vectors in the feature space. Thus, using the kernel function one only has to calculate the inner products between support vectors and the vectors in the input space [4],[75]. This solves the technical difficulties encountered by learning machines in high-dimensional spaces [10],[11],[39] and consequently the curse of dimensionality with respect to the computational performance is overcome.

3.4 Types of kernels

Many of the characteristics of the model in (3.5) are determined by the type of kernel function being used [88]. Which kernel function to use depends on many factors such as the type of problem that is being solved, the unknown underlying functional dependency, the type and number of data points, the suitability for online and off-line learning, the computational resources and experience or expertise available for a specific kernel [39]. Some kernel functions are better equipped to solve certain problems than others. For example, if a problem is highly nonlinear in the input space, one should use a kernel function that can cope with that. One can use a simple inner product kernel which produces a linear mapping or more complicated kernels such as β -spline kernels and Fourier Transform kernels.

Recall that in order to guarantee that the kernel function has the expansion in (3.1), it must satisfy Mercer's conditions as stated in (3.3). These conditions are used to build various types of kernels. Since in practice we will only use finite dimensional learning data, an admissible kernel function must result in a symmetric, positive semi-definite matrix. From Hilbert-Schmidt theory, we also know that inner products define metrics or distance measures in inner product spaces [42]. Different distance measures result in different types of kernels. Numerous possibilities of kernels exist and it is often difficult to explain their individual characteristics. However, there are two main types of kernels, namely *local* and *global* kernels [86]. In local kernels only the data that are close or in the proximity of each other have an influence on the kernel values. In contrast, a global kernel allows data points that are far away from each other to have an influence on the kernel values as well. For more information on the characteristics of various kernels see [88].

An example of a typical local kernel is the RBF kernel,

$$K(\mathbf{x}, \hat{\mathbf{x}}) = \exp \left\{ -\frac{|\mathbf{x} - \hat{\mathbf{x}}|^2}{2\sigma^2} \right\}, \quad (3.6)$$

where the kernel parameter, σ , is the width of the radial basis function. The polynomial kernel, a typical example of a global kernel, is defined as

$$K(\mathbf{x}, \hat{\mathbf{x}}) = [\langle \mathbf{x}, \hat{\mathbf{x}} \rangle + 1]^q, \quad (3.7)$$

where the kernel parameter q is the degree of the polynomial to be used.

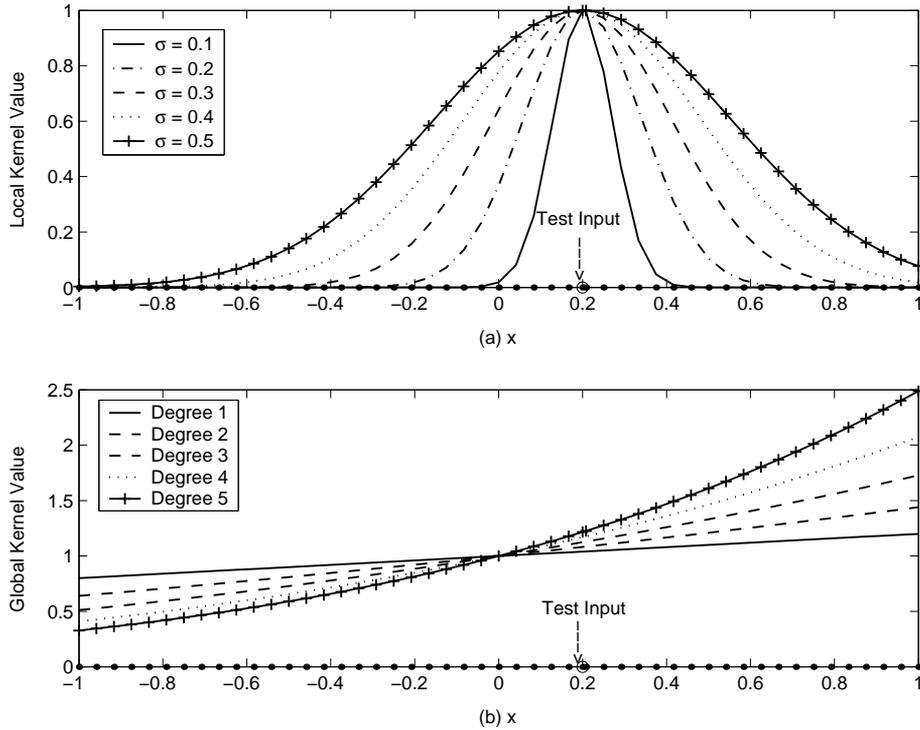


Figure 3.3: Examples of (a) a local kernel (RBF) and (b) a global kernel (polynomial).

The behaviour of the two types of kernels is shown in Figure 3.3 where for linearly distributed data over $[-1, 1]$ the kernel values with respect to a specific test point at 0.2 are determined. In Figure 3.3(a) the local effect of the RBF kernel is shown for the chosen test input, for different values of the width σ . One can clearly see that at some point, the kernel values level off to zero. A local kernel therefore only has an effect on the data points in the neighbourhood of the test point. In Figure 3.3(b), the global effect of the polynomial kernel of various degrees can be seen. Consider the same test data point as used in the case of the local kernel. For each degree of the polynomial, all data points in the input domain have non-zero kernel values. The test data point has a global effect on the other data points. This can be explained by noting that in (3.7) every data point from the set \mathbf{x} has an influence on the kernel value of the test point $\hat{\mathbf{x}}$, irrespective of its the actual distance from $\hat{\mathbf{x}}$.

Many different kernel functions can be constructed as long as they satisfy Mercer's Conditions in (3.3). Kernel functions can also be constructed from other kernels or from features. See [10], [88] and [93] for more information. We use in Chapter 5 a mixture of a RBF kernel and a polynomial kernel.

To visually explain the use of the kernel on the SVM, consider a two-dimensional data set X over $[0, 1] \times [0, 1]$ of ℓ . A test set \hat{X} of 25 points is constructed from a grid $[0, 1] \times [0, 1]$. Two $(\ell \times 25)$ kernel matrices are determined using an RBF kernel with $\sigma = 0.2$ and a polynomial kernel with $q = 2$. One column of the kernel matrix then contains the kernel values of data points in X with respect to one of the test data points $\hat{\mathbf{x}}_j \in \hat{X}$. Assign to entry i of the column vector the triple $(x_{i1}, x_{i2}, K(\mathbf{x}_i, \hat{\mathbf{x}}_j))$. For each test point the triples are then plotted 3D. In Figure 3.4 the kernel values of an RBF kernel for each of the 25 test points were plotted. The same was done in Figure 3.5 using a polynomial kernel.

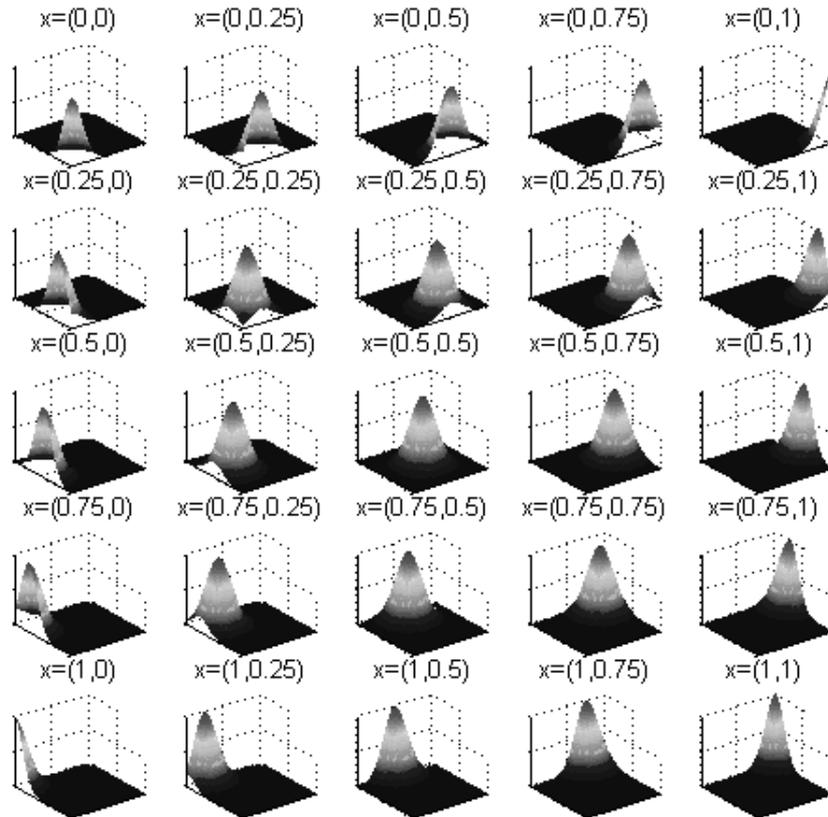
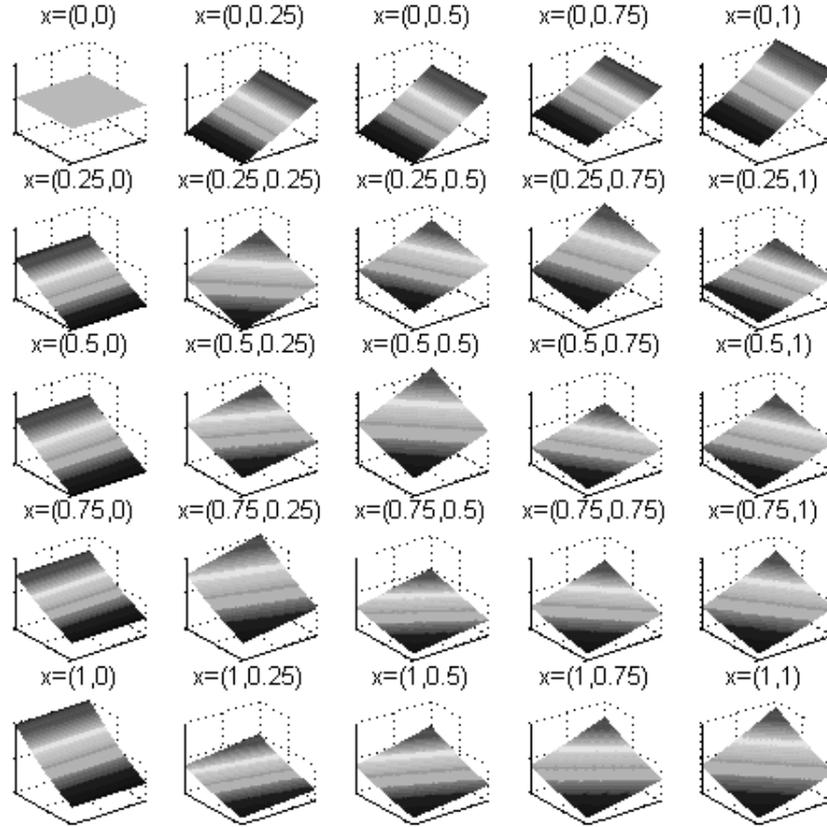


Figure 3.4: RBF kernel with $\sigma = 0.2$.

Figure 3.5: Polynomial kernel with *degree* = 2.

One can see that the behaviour of the RBF kernel in Figure 3.4 is quite different from that of the polynomial kernel in Figure 3.5. Each column of the kernel matrix $K(X, \hat{X})$ is represented by one of the planes in Figures 3.4 and 3.5. For a data set of ℓ samples, the optimisation problem in (3.4) would create ℓ planes and determine the weight for each plane. The vector of the predictions of the original ℓ learning data points then results in a weighted linear combination of the planes corresponding to the support vectors,

$$\mathbf{y} = \left[\sum_{i \in SV} \omega_i K(\mathbf{x}_i, X) \right] + b \mathbf{1}_\ell,$$

where $K(\mathbf{x}_i, X)$ is a column vector of length ℓ and $\mathbf{1}_\ell$ is an $(\ell, 1)$ vector of ones.

3.5 Choice of kernel and kernel parameters

The type of kernel is the most important choice to be made, since it influences very important characteristics of the model. These characteristics include interpolation and extrapolation ability, robustness and scaling. There are many kernels that can be used. The research has focussed on the RBF kernels and polynomial kernels since these are the more commonly used and most of the problems encountered can be solved by either of the two or a mixture of the two. See Chapter 5 for more information on the mixed kernel approach. Apart from the choice of kernel, the kernel parameter(s) must also be set or optimised if necessary.

3.5.1 Radial Basis Function kernel

This type of kernel is classified as a local kernel and it gives the SVM model good interpolation properties. The kernel function has the ability to detect local phenomena. However, the nature of the RBF kernel is such that the resulting model will fail to extrapolate outside the known input space because the kernel values level off once unknown domains of the input space are entered. This needs not be a disadvantage though, for under certain circumstances such a behaviour may be preferable.

In data sets containing a significant number of outliers or regions of low data density, the characteristic of detecting local phenomena may be problematic. For example, in Figure 3.6 the behaviour of an RBF kernel model is shown for three data sets that vary in noise, presence of outliers and data density. For all three data sets the same SVM parameters were used, namely $C = 100,000$ and $\epsilon = 0.1$. Figure 3.6(a) shows that the model performs quite well for a moderate noise level. In Figure 3.6(b), however, the RBF kernel picks up on certain local abnormalities and the model reflects that. Fortunately, the effect is far less than what one would normally expect. In Figure 3.6(c), on the other hand, the negative effect of varying data density is more evident.

How much the local phenomena influence the model, is determined by the width of the bases. Using too small a width will result in a learning machine that tends to model noise. Using too large a width will result in a learning machine that may ignore important local behaviour. These effects can be seen in Figure 3.7 where the different values of σ are used.

The value of the width σ obviously depends on the scale of the data. Also note that the same width is used for all features. To avoid having kernel values dominated by a single feature, it is better to scale each feature of the input data to the same range. In industry, input data are therefore often range scaled such that the minimum and maximum of each input dimension are 0 and 1 respectively. Another type of scaling also frequently used is mean scaling where each input dimension is scaled such that it has zero mean and unit variance.

From our experience the RBF kernel is a good kernel to start with when analysing a data set in order to find the optimal settings of the other parameters such as the complexity and regularisation parameters. Optimisation of the value of σ may be done through cross-validation [55],[58],[60], any line search method [6] or inspection of a range of values [39],[60]. Since cross-validation techniques can be very time consuming

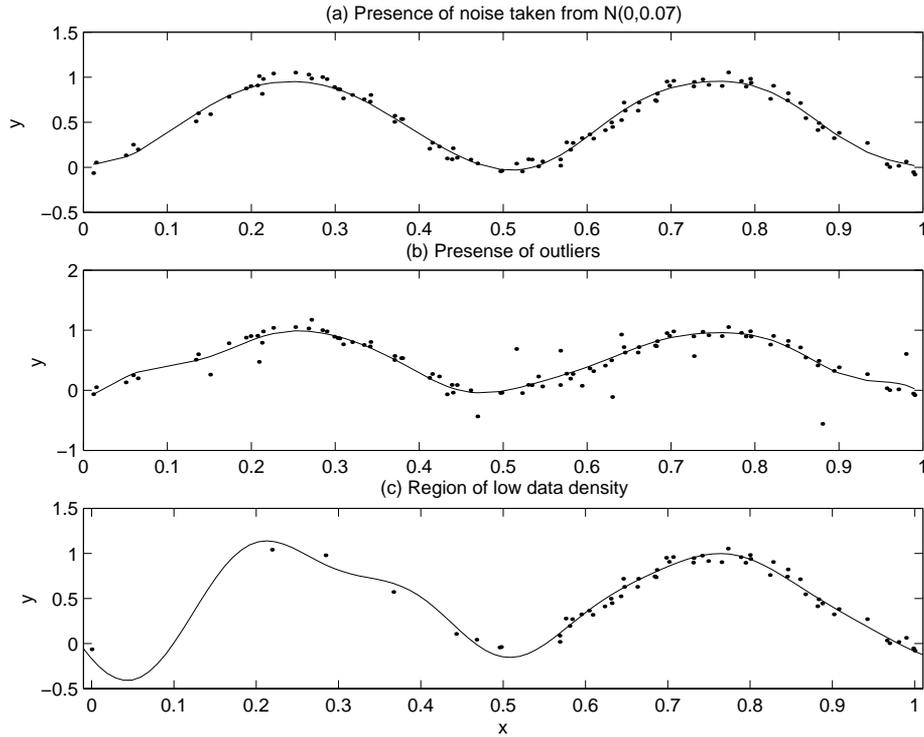
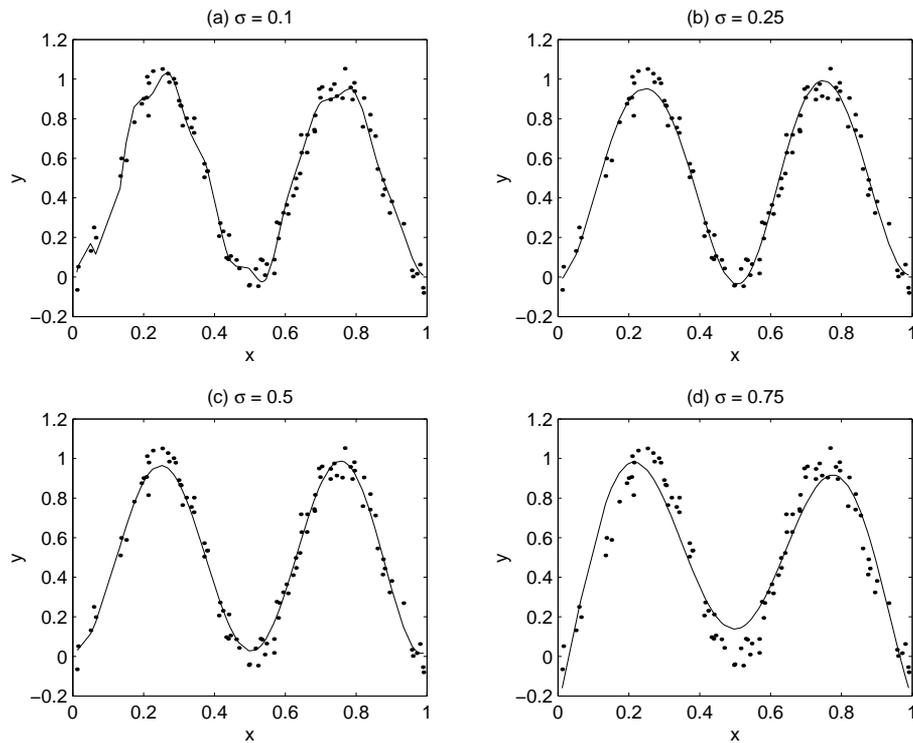


Figure 3.6: The behaviour of SVM models using an RBF kernel for different data sets. Graph (a) shows a data set with moderate noise level. In Graph (b) a number of outliers were added. Graph (c) contains a data set with a low data density in $[0, 0.5]$.

and are often not suitable for smaller data sets [55], we chose to implement the latter approach in the software. The parameter that results in a model with the lowest estimated VC-dimension [106],[11],[39] is then selected. Alternatively, one could also use Vapnik's measure, an estimation of the theoretical prediction risk [55],[11],[60] as a selection criterion. Vapnik's measure is given in Chapter 9. In Algorithm E the algorithmic pseudo code for selecting the kernel parameter is given.

Algorithm E. Optimise kernel parameter

- Step 1:** Select the learning data $(\mathbf{x}_i, y_i), i = 1, \dots, \ell$, kernel function K , C , a complexity parameter (ϵ or ν) and iteration parameters ker_start, ker_end and it_num .
- Step 2:** Construct a vector \mathbf{p} consisting of it_num values linearly distributed ranging from ker_start to ker_end .

Figure 3.7: The effect of σ on the SVM model.

Step 3: For $k = 1$ to it_num
 if complexity parameter is ϵ ,
 run Algorithm B using $p(k)$,
 else run Algorithm C using $p(k)$.
 set $\mathbf{w}(k) = \mathbf{w}$
 set $b(k) = b$
 $SV(k) = \{SV\}$
 construct the model $f_k(\hat{\mathbf{x}}) = \mathbf{w}(k)^T K_{p(k)}(\mathbf{x}, \hat{\mathbf{x}}) + b(k)$
 calculate the predictions $\mathbf{py}(k) = f_k(\mathbf{x})$
 end

Step 4: Find optimal model f^* using Vapnik's measure.

Step 5: Return optimal kernel parameter $ker = p^*$.

3.5.2 Polynomial kernel

This type of kernel is classified as a global kernel and it gives the SVM model good extrapolation properties. It may fail to interpolate well, since local phenomena have little influence on the kernel values and therefore, it is less sensitive to outliers. However, when these local phenomena are not due to noise or outliers, it may be better to choose a local kernel like the RBF kernel. In Figure 3.8 the behaviour of a polynomial

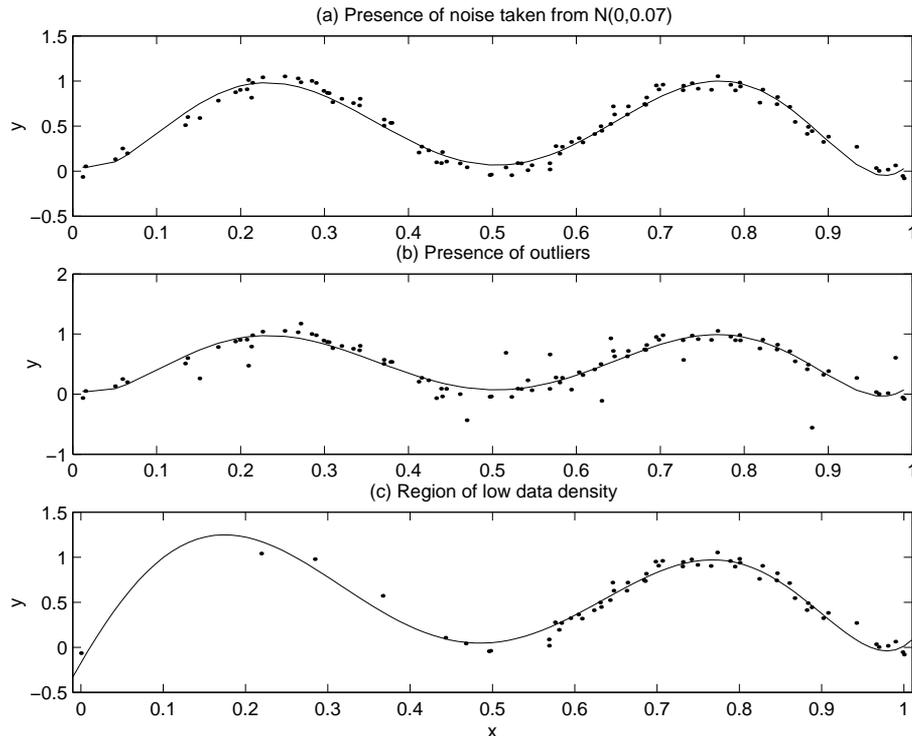


Figure 3.8: The behaviour of SVM models using a seventh degree polynomial kernel for different data sets. Graph (a) shows a data set with moderate noise level. In Graph (b) a number of outliers were added. Graph (c) contains a data set with a low data density in $[0, 0.5]$.

kernel model is shown for a data set with moderate noise, a data set with outliers and a data set with a region of low data density. In all three graphs of Figure 3.8 the same SVM parameters ($C = 100.000, \epsilon = 0.1$) were used. From Figure 3.8 one can see that the polynomial kernel is less sensitive to noise and outliers. Furthermore, since the polynomial kernel has better extrapolation properties, it is less vulnerable for regions where the data density is lower than the RBF kernel.

The choice of the degree of the polynomial kernel should be taken with great care. In many nonlinear problems a rather high degree of polynomial kernel (above ten) may

be needed. In Figure 3.9 several degrees are used. Note that by increasing the degree

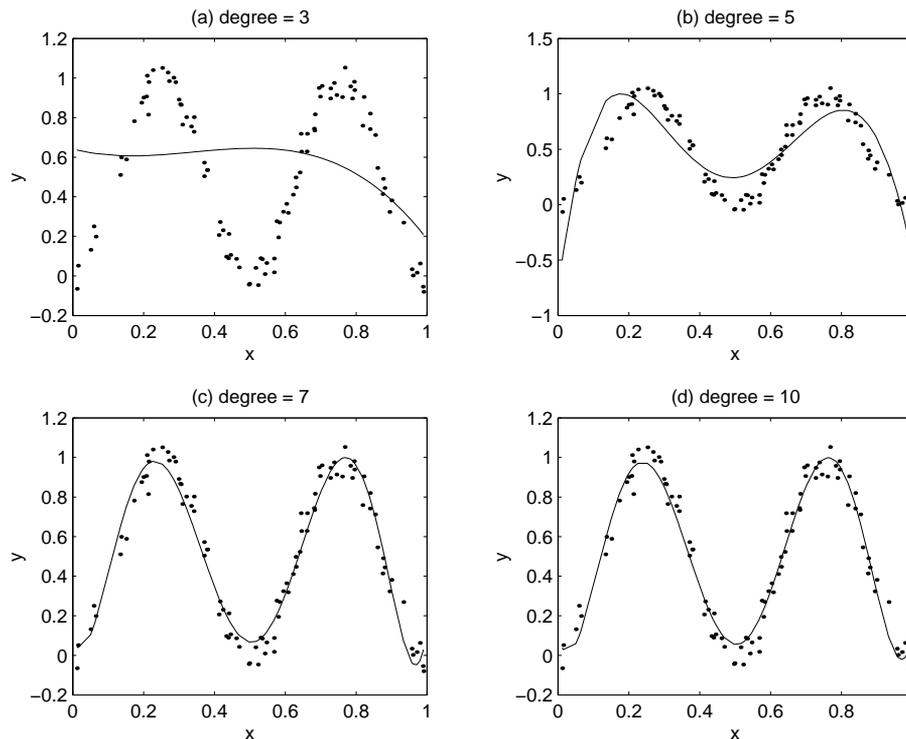


Figure 3.9: The effect of the degree on the SVM model.

of the polynomial it may appear that a better model is built since the interpolation ability increases. However, as will be shown in Chapter 5, the extrapolation ability decreases dramatically at the same time, because high degrees of the polynomial may cause the kernel values to explode. To avoid these runaway kernel values, it is better to scale the input data and avoid using polynomials of too high a degree [88].

From our experience polynomial kernels should only be used when one suspects the model to be polynomial of a relatively low degree. Optimisation of the degree is similar to that of the width of the RBF kernel. The only difference with algorithmic pseudo code given in Algorithm E is that in Step 2 only integer values are allowed.

3.6 Conclusion

Let us state the original design requirement again:

The inferential sensor should be able to handle high-dimensional data and overcome the curse of dimensionality.

One of the main advantages of using SVMs is that a kernel approach can be used such that the learning machine partially overcomes the curse of dimensionality. We say *partially* since the SVM only overcomes the part with respect to the predictive performance to some extent. Since the SVM measures the complexity of the model in terms of number of observations and the VC-dimension rather than the dimensionality of the feature space, the curse of dimensionality with respect to making local estimates, given in Properties 1 and 2, is overcome. However, the problem that the prediction of unseen data points often results in extrapolation (Properties 3 and 4) remains. This problem can only be solved with extra data or the addition of prior knowledge of the unknown domains in the input space.

The second part of the curse of dimensionality, the computational degradation, is overcome through the use of kernel functions since the very high-dimensional feature spaces are not considered explicitly but implicitly. It is now possible to use very high-dimensional learning data without running into computational difficulties. Furthermore, the support vector machine method is ideally formulated to use various kinds of kernel functions in order to adapt to a nonlinear problem. Very complex and nonlinear kernels can be used, resulting in highly nonlinear models in the input space.

One precautionary remark: It would be wrong to think that the Support Vector Machine method makes feature selection applications obsolete. The SVM still has the underlying assumption that *all* dimensions used, are relevant. In many real-life data, especially in process data, the data set could be corrupted with numerous dimensions that are irrelevant. Some of these irrelevant features may even consist only of noise. A rule of thumb in this regard is *garbage in, garbage out*. Therefore, feature selection should still be performed as a preprocessing step in order to remove all irrelevant dimensions.

Chapter 4

Robustness

4.1 Introduction

Inferential sensors often have to operate with data that contain noise and several outliers, and with the possibility that all kinds of changes may occur in the plant. The term *robustness* is often used in industry to describe the inferential sensor's sensitivity to perturbations in the variables, parameters or learning data [22].

One important issue in industrial data is the presence of noise as well as the characteristics of the noise. The noise is often neither constant nor normally distributed, which is an assumption many learning methods make. It is also known that the quadratic loss function, which is used by least squares methods, is only optimal for Gaussian noise models, i.e. normally distributed noise [22]. When the noise is not Gaussian then, under the assumptions of symmetry and convexity of the noise probability function, the least modulus loss,

$$L(y, f(\mathbf{x}, \alpha)) = |y - f(\mathbf{x}, \alpha)|, \quad (4.1)$$

should be used [27]. Minimisation of the empirical risk using (4.1) defines the *robust regression* function [106]. In particular when only general information about the noise is known, as is often the case, it is better to use the least modulus loss [22],[39].

Another issue is the presence of outliers. They occur all too frequently, even in carefully conducted experiments [71]. In highly non-linear data sets the input and output values of the outliers are most likely within the range of the input and output space variables. As most classical outlier detection tools use graphical presentation as detection method, the detection of such outliers will often fail [33]. The consequence of this is that a number of outliers may still be present during the learning phase. Therefore, it is necessary that the inferential sensor has some level of tolerance to outliers [26].

It has been observed that one characteristic aspect of any inferential sensor is that it should not only make accurate predictions, but also be robust to moderate changes in the data [22]. Therefore, in the development of a inferential sensor it is required that the learning machine resolves the subtle trade-off between accuracy and robustness. In SVM for regression this trade-off is obtained through the use of

different loss functions [78]. Furthermore, as the regularisation parameter C controls the trade-off between the accuracy and the complexity [106], it is necessary to obtain some insight into the use of this parameter.

In the first section of this chapter we show that the ϵ -insensitive loss function is a realisation of the least modulus loss. In the second section the SVM for regression using a quadratic ϵ -insensitive loss function is given and the algorithmic pseudo codes for the ϵ -SVM and ν -SVM using the quadratic ϵ -insensitive loss are given. Finally in the third section the effect of the regularisation parameter is discussed and a method for estimating its value is derived.

4.2 Linear ϵ -SVM for regression

Consider the ϵ -SVM discussed in Chapter 2

$$\text{minimise} \quad \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{\ell} \sum_{i=1}^{\ell} (\xi_i + \xi_i^*), \quad (4.2a)$$

$$\text{subject to} \quad (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - y_i \leq \epsilon - \xi_i, \quad i = 1, \dots, \ell, \quad (4.2b)$$

$$y_i - (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \leq \epsilon + \xi_i^*, \quad i = 1, \dots, \ell, \quad (4.2c)$$

$$\xi_i, \xi_i^* \geq 0, \quad i = 1, \dots, \ell. \quad (4.2d)$$

Recall that through introducing Lagrange multipliers and finding the saddle point of the Lagrangian, the Wolf dual problem can be written as

$$\begin{aligned} \text{maximise} \quad & \sum_{i=1}^{\ell} y_i (\alpha_i^* - \alpha_i) - \epsilon \sum_{i=1}^{\ell} (\alpha_i^* + \alpha_i) \\ & - \frac{1}{2} \sum_{i,j=1}^{\ell} (\alpha_i^* - \alpha_i) (\alpha_j^* - \alpha_j) K(\mathbf{x}_i, \mathbf{x}_j), \end{aligned} \quad (4.3a)$$

$$\text{subject to} \quad \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) = 0, \quad (4.3b)$$

$$0 \leq \alpha_i, \alpha_i^* \leq \frac{C}{\ell}, \quad i = 1, \dots, \ell. \quad (4.3c)$$

Defining $\omega = \alpha^* - \alpha$ and SV as the set of support vectors, the resulting model then has the form

$$f(\hat{\mathbf{x}}) = \sum_{i \in SV} \omega_i K(\mathbf{x}_i, \hat{\mathbf{x}}) + b, \quad (4.4)$$

where b is determined by (2.53).

In Chapter 2 we explained that the linear ϵ -insensitive loss function has the advantage of realising a model that has a sparse representation in terms of the support vectors, as seen in (4.4). However apart from that, the ϵ -insensitive zone has another advantage. For $\epsilon = 0$ the ϵ -insensitive loss is equivalent to the least modulus loss [39],[88]. Thus the SVM for regression using the linear ϵ -insensitive loss function is suitable for problems for which only general information on the noise is available.

Furthermore, as it is known that there is a linear dependency between ϵ and the amount of noise in the data [45],[90], the following observations can be made with respect to the influence of noise on the performance of the SVM [59],[55]:

1. If the noise power is very small compared to ϵ^2 the performance of the SVM for regression is not affected by the presence of noise;
2. If the noise power is smaller than ϵ^2 the ϵ -insensitive zone masks the presence of noise and the performance of the SVM can be expected to increase;
3. If the noise power is equal to or larger than ϵ^2 the ϵ -insensitive zone no longer mask the noise and the ratio of support vectors saturates to one. The performance of the SVM slowly degrades as the ratio of support vectors approaches one.

Therefore, ϵ is an ideal parameter for controlling the effect that noise has on the performance of the model.

Note that the ϵ -insensitive loss function allows only data points that are outside the insensitive zone to enter the empirical risk term in (4.2a). It would therefore seem that in the presence of outliers the SVM for regression is largely determined by the outliers. The contrary is in fact true as seen in Figure 4.1. It has been proven in [78] that, using the SVM for regression with the linear ϵ -loss function, local movements of the output values of data points outside the ϵ -insensitive zone do not influence the regression. The proof is based on the fact that for all data points outside the insensitive tube, the upper bound for the corresponding α_i or α_i^* in (4.3c) is the same. In fact, all support vectors that do not touch the tube will have an absolute value equal to the upper bound. The result is that the outliers do not have an adverse effect on the model. Note that one of the requirements for robust estimators is that the influence of the outlier data points should be bounded from above [78], [28]. Thus, the resistance of the SVM for regression to outliers is in close spirit with robust estimators, making it ideal for industrial applications [22].

4.3 Quadratic ϵ -SVM for regression

We argued that the linear ϵ -insensitive loss should be used when the noise distribution of the data is not expected to be Gaussian and especially in cases where outliers might still be present in the data. There are of course cases where one might prefer to use the quadratic loss function, but with the advantage of complexity control and insensitivity to low noise levels [58]. To construct such a learning machine in the spirit of support vector theory, the quadratic ϵ -insensitive loss function, as seen in Figure 4.2, can be used. For a given learning or training data set $\mathbf{x}_i \in X \subseteq \mathbb{R}^n, y_i \in \mathbb{R}$ with ℓ , the quadratic ϵ -insensitive loss function is formally given by

$$L_2^\epsilon(\mathbf{x}, y, f(\mathbf{x})) = |y - f(x)|_\epsilon^2. \quad (4.5)$$

The ϵ -SVM for regression using the quadratic ϵ -insensitive loss function is then given

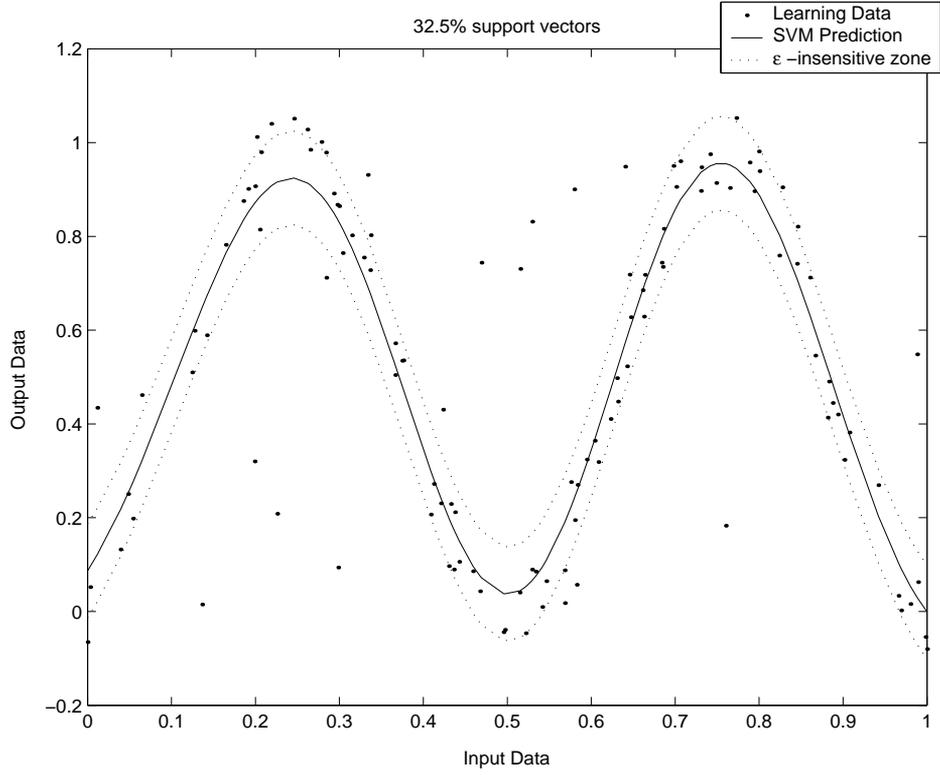


Figure 4.1: The quadratic ϵ -insensitive loss-function.

by

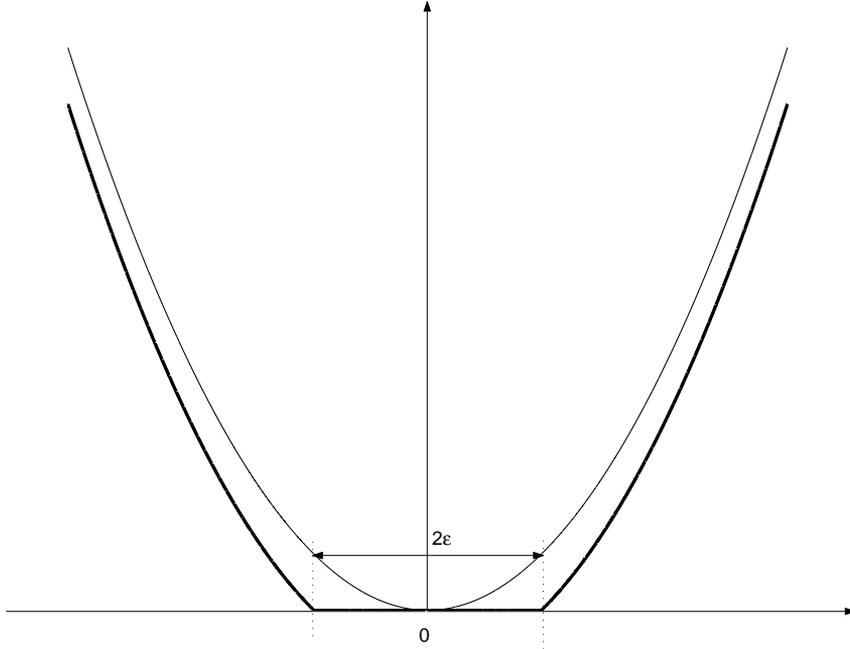
$$\text{minimise} \quad \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{\ell} \sum_{i=1}^{\ell} (\xi_i^2 + (\xi_i^*)^2), \quad (4.6a)$$

$$\text{subject to} \quad (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - y_i \leq \epsilon + \xi_i, \quad i = 1, \dots, \ell, \quad (4.6b)$$

$$y_i - (\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \leq \epsilon + \xi_i^*, \quad i = 1, \dots, \ell, \quad (4.6c)$$

$$\xi_i, \xi_i^* \geq 0, \quad i = 1, \dots, \ell. \quad (4.6d)$$

Similarly as in Chapter 2, the dual formulation of the QP problem in (4.6) can be

Figure 4.2: The quadratic ϵ -insensitive loss-function.

obtained. After using the kernel “trick” again, the Wolf dual can be written as

$$\begin{aligned} \text{maximise} \quad & -\frac{1}{2} \sum_{i,j=1}^{\ell} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) \left(K(\mathbf{x}_i, \mathbf{x}_j) + \frac{\ell}{2C} \delta_{i,j} \right) \\ & + \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) y_i - \epsilon \sum_{i=1}^{\ell} (\alpha_i^* + \alpha_i), \end{aligned} \quad (4.7a)$$

$$\text{subject to} \quad \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) = 0, \quad (4.7b)$$

$$\alpha_i \geq 0, \alpha_i^* \geq 0, \quad i = 1, \dots, \ell. \quad (4.7c)$$

The SVM model, in terms of the support vector weights $\boldsymbol{\omega} = \alpha^* - \alpha$, remains the same as in (4.4).

Next we write the QP problem in terms of the Hessian matrix and separate matrices for the equality and inequality constraint. Define I_n as the $n \times n$ identity matrix. The Hessian matrix \mathbf{H} is the $2\ell \times 2\ell$ matrix such that

$$H = \begin{pmatrix} K(\mathbf{x}, \mathbf{x}) & -K(\mathbf{x}, \mathbf{x}) \\ -K(\mathbf{x}, \mathbf{x}) & K(\mathbf{x}, \mathbf{x}) \end{pmatrix} + \frac{\ell}{2C} I_{2\ell}.$$

For $\beta = \begin{bmatrix} \alpha^* \\ \alpha \end{bmatrix}$, the SVM for regression in matrix notation is then to minimise

$$\frac{1}{2}\beta^T \mathbf{H}\beta + \begin{bmatrix} \epsilon \mathbf{1}_\ell - \mathbf{y} \\ \epsilon \mathbf{1}_\ell + \mathbf{y} \end{bmatrix}^T \beta \quad (4.9)$$

subject to the equality constraint

$$\begin{bmatrix} \mathbf{1}_\ell \\ -\mathbf{1}_\ell \end{bmatrix}^T \beta = 0 \quad (4.10)$$

and the inequality constraints

$$\mathbf{0}_{2\ell} \leq \beta. \quad (4.11)$$

The algorithmic pseudo code of the ϵ -SVM for regression with quadratic loss can be found in Algorithm F.

Algorithm F. ϵ -SVM for regression with quadratic loss

-
- Step 1:** Select the learning data $(\mathbf{x}_i, y_i), i = 1, \dots, \ell$, kernel function K , C and *epsilon*.
- Step 2:** Construct matrices for the QP problem.
- Step 3:** Solve (4.9) subject to (4.10) and (4.11) for β .
- Step 4:** Steps 4-10 of Algorithm B.
-

Similarly to the derivation in Chapter 2, the ν -SVM with quadratic ϵ -insensitive loss function can be given. We give only the optimisation problem in terms of the required matrices. The Hessian matrix \mathbf{H} is the same as in (4.8). Using $\beta = \begin{bmatrix} \alpha^* \\ \alpha \end{bmatrix}$, the ν -SVM for regression using quadratic loss minimises

$$\frac{1}{2}\beta^T \mathbf{H}\beta + \begin{bmatrix} -\mathbf{y} \\ \mathbf{y} \end{bmatrix}^T \beta \quad (4.12)$$

subject to the equality constraint

$$\begin{bmatrix} \mathbf{1}_\ell \\ -\mathbf{1}_\ell \end{bmatrix}^T \beta = 0 \quad (4.13)$$

and the inequality constraints

$$\begin{bmatrix} \mathbf{0}_{2\ell} \\ 0 \end{bmatrix} \leq \begin{bmatrix} \beta \\ (\mathbf{1}_{2\ell})^T \beta \end{bmatrix} \leq \begin{bmatrix} \infty \cdot \mathbf{1}_{2\ell} \\ C \cdot \nu \end{bmatrix}. \quad (4.14)$$

The algorithmic pseudo code of the ν -SVM for regression with quadratic loss can be found in Algorithm G.

Algorithm G. ν -SVM for Regression with quadratic loss

- Step 1:** Select the learning data $(\mathbf{x}_i, y_i), i = 1, \dots, \ell$, kernel function K , C and ν .
- Step 2:** Construct the matrices for the QP problem.
- Step 3:** Solve (4.12) subject to (4.13) and (4.14) for β .
- Step 4:** Steps 4-10 of Algorithm B.
-

4.4 Choice of regularisation parameter

Consider the ϵ -SVM with quadratic ϵ -insensitive loss in (4.6). The parameter C in (4.6a) controls the trade-off between the complexity of the model ($1/2 \|\mathbf{w}\|^2$) and the training error ($1/\ell \sum_{i=1}^{\ell} (\xi_i^2 + (\xi_i^*)^2)$) [106]. C is also called the regularisation parameter since it corresponds to the parameter γ of the regularisation method for solving ill-posed problems as $C = 1/\gamma$ [108].

Finding the optimal value for C still remains a problem. Many researchers suggested that C should be varied through a wide range of values and the optimal performance is then measured by using a separate validation set or other techniques such as cross-validation or boot-strapping [10],[55],[58],[60]. Vapnik mentioned in [106] three methods for choosing the optimal regularisation parameter, namely, the L-curve method [15], the method of effective number of parameters [24] and the effective VC-dimension method [108],[109]. Each of these methods uses a different approach for measuring the performance and complexity of the model and originates from different theories. One common problem with many of the suggested approaches is that they are not suitable for large-scale problems. The computational effort to determine the eigenvalues of large matrices or using resampling limits their use in online applications.

In particular, if one needs to make a quick assessment whether a given data set can be solved with the SVM method or if a given kernel function is an appropriate choice, a fast estimation method is extremely useful. Furthermore, since the C parameter is known to be a rather robust parameter, determining the true optimal value is often not worth the effort [34]. In SVM literature it is often suggested that C should be chosen *sufficiently large*. But what value is large enough? If an estimation method can give a good indication of the magnitude of C , one can at least start from an informed guess.

It is known that the scale of the regularisation parameter is affected by several factors. It has been shown by Smola [87] that the optimal regularisation parameter depends on the value ϵ . Since ϵ is used to control the complexity of the model and depends on the noise level in the data, the choice of the optimal value of C assumes some knowledge about the underlying noise distribution as well as the inherent complexity of the model. Often, this knowledge is not available. In [10] the authors indicate that the regularisation parameter C is also affected by the choice of feature space. The

consequence of this is very significant, since the feature space is determined by the specified kernel, which is in fact an operator associated with smoothness. Therefore, the choice of regularisation parameter cannot be based on one factor alone, but on the combined influence. None of the heuristics of estimation methods in the literature does that. The research was therefore aimed at deriving an estimating rule that combines the characteristics of the feature space, the expected noise level, and some other contributing factors.

4.4.1 Results from the L-curve method

The L-curve method is derived from the theory of solving ill-posed problems [15]. It is a well established method and one of the few approaches in regularisation theory that takes into account both the norm of the solution and the norm of the error [23]. Vapnik has shown in [106] that the L-curve Method can be applied for SVMs for regression with a quadratic loss function. The resulting terms for the norm of the solution and the norm of the error, are then given by the following two functions,

$$\begin{aligned}\eta(\gamma) &= \|\mathbf{w}_\gamma\|^2, \\ &= \sum_{i,j=1}^N \beta_i(\gamma)\beta_j(\gamma)K(\mathbf{x}_i, \mathbf{x}_j)\end{aligned}\quad (4.15)$$

and

$$\begin{aligned}\rho(\gamma) &= \frac{1}{\ell} \sum_{i=1}^{\ell} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 \\ &= \frac{1}{\ell} \sum_{i=1}^{\ell} \left(y_i - \sum_{k=1}^N \beta_k(\gamma)K(\mathbf{x}_k, \mathbf{x}_i) \right)^2,\end{aligned}\quad (4.16)$$

where N is the index set of the support vectors.

The L-curve is the log-log plot of $\eta(\gamma)$ against $\rho(\gamma)$. The distinct L-shape of the curve is shown in Figure 4.3. The L-curve method is a very useful graphical tool which is used to display the trade-off between the complexity and the error. If too little complexity is used, the right ‘leg’ of the L-curve is dominant and the model typically underfits (see Figure 4.3(b)). When the left ‘leg’ of the L-curve is dominant, the model uses too much complexity and starts to overfit as seen in Figure 4.3(d). The corner point of the L-curve corresponds to the optimal value of the regularisation parameter for which the model has the right balance between complexity and the error term.

Finding the corner point of the L-curve involves finding the minimum of the func-

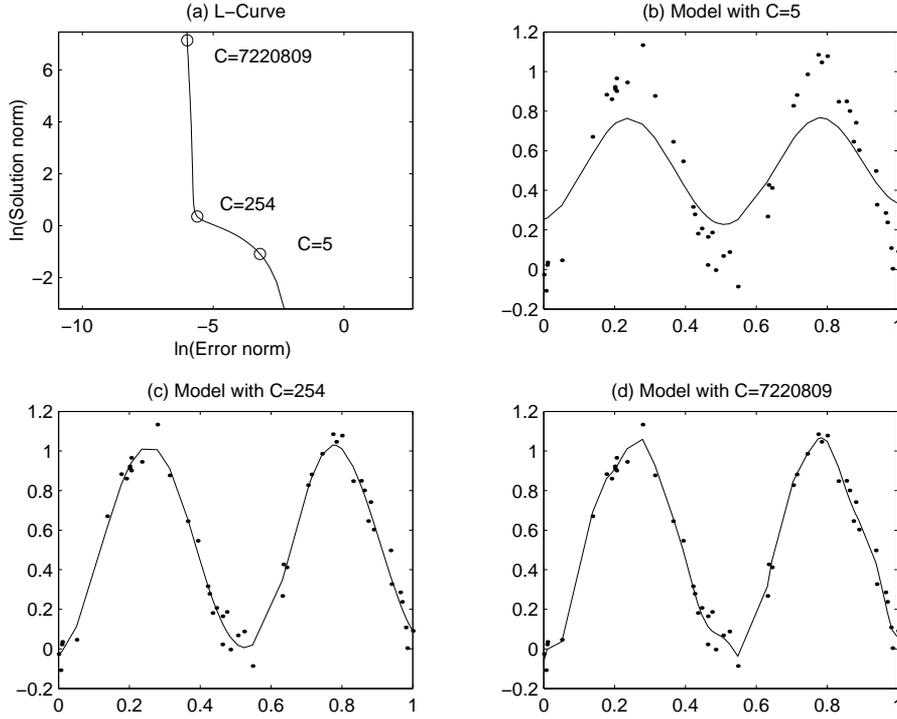


Figure 4.3: The form of the L-curve is shown in graph (a). Graphs (b),(c) and (d) show models for various values of C .

tional

$$\begin{aligned}
 H(\gamma) &= \frac{1}{\ell} \sum_{i=1}^{\ell} \left(y_i - \sum_{k=1}^N \beta_k(\gamma) K(x_k, x_i) \right)^2 \\
 &\quad \sum_{i,j=1}^N \beta_i(\gamma) \beta_j(\gamma) K(x_i, x_j) \\
 &= \rho(\gamma) \eta(\gamma).
 \end{aligned} \tag{4.17}$$

In regularisation theory, the corner point of the L-curve is normally found by determining the curvature of the L-curve. In [23] an expression for the curvature of the L-curve is derived in terms of $\rho(\gamma)$ and $\eta(\gamma)$ and their derivatives. As part of the derivation of the curvature expression, an important relation between the derivative of $\rho(\gamma)$ and $\eta(\gamma)$ emerged. And it is this relation we are interested in.

Consider the following minimization problem

$$z_\gamma = \arg \min \left\{ \|Az - b\|_2^2 + \gamma \|z_\gamma\|_2^2 \right\}, \tag{4.18}$$

where A is a symmetric positive (semi)definite coefficient matrix and b the given output data. Using the SVD decomposition of A [97], the norms of the solution and error can be written as

$$\eta(\gamma) = \sum_{i=1}^{\ell} \left(f_i \frac{u_i^T b}{\sigma_i} \right)^2 \quad (4.19)$$

$$\text{and } \rho(\gamma) = \sum_{i=1}^{\ell} ((1 - f_i) u_i^T b)^2, \quad (4.20)$$

where u_i are the singular vectors, σ_i the singular values, and f_i the Tikhonov filter factors [21], that depend on σ_i and γ as follows,

$$f_i = \frac{\sigma_i^2}{\sigma_i^2 + \gamma}. \quad (4.21)$$

The derivatives of $\eta(\gamma)$ and $\rho(\gamma)$ to γ , are then given by

$$\eta'(\gamma) = -\frac{2}{\gamma} \sum_{i=1}^{\ell} (1 - f_i) f_i^2 \left(\frac{u_i^T b}{\sigma_i} \right)^2 \quad (4.22)$$

$$\text{and } \rho'(\gamma) = \frac{2}{\gamma} \sum_{i=1}^{\ell} f_i (1 - f_i)^2 (u_i^T b)^2. \quad (4.23)$$

(Note that in [23], these equations were derived using γ^2 which resulted in having a factor 4 instead of 2 in each equation.) Rewriting $\rho'(\gamma)$ and using the fact that

$$\frac{f_i}{\sigma_i^2} = \frac{1}{\sigma_i^2 + \gamma} = \frac{1 - f_i}{\gamma},$$

leads to the following important relation

$$\rho'(\gamma) = -\gamma \eta'(\gamma). \quad (4.24)$$

4.4.2 Estimate for C

The relation (4.24) also applies to the Support Vector Machine formulation with quadratic loss when the implicit feature space, defined by the kernel, is considered. In this section, the relation (4.24) combined with (4.17), is used to derive an estimate¹. First, consider the functional (4.17). In order to find the optimal regularisation parameter γ , (4.17) has to be minimized, that is to set $H'(\gamma) = 0$. The derivative of $H'(\gamma)$ is given by

$$\begin{aligned} \rho'(\gamma)\eta(\gamma) + \rho(\gamma)\eta'(\gamma) &= 0 \\ \text{and } \frac{\rho'(\gamma)}{\eta'(\gamma)} &= \frac{\rho(\gamma)}{\eta(\gamma)}. \end{aligned} \quad (4.25)$$

¹It is also interesting to note the close resemblance between the derivation of the expression for the curvature of the L-curve, which uses the SVD decomposition, and the use of the eigenvalues and eigenvectors in the method of the effective number of parameters that was suggested in statistics for estimating parameters for ridge regression [24].

Rewriting the relation (4.24), given in the previous section, such that γ stands alone and using (4.25), leads to

$$\gamma = -\frac{\rho'(\gamma)}{\eta'(\gamma)} = \frac{\rho(\gamma)}{\eta(\gamma)}.$$

Now using the fact that $C = 1/\gamma$, we arrive at

$$C = \frac{\eta(\gamma)}{\rho(\gamma)}. \quad (4.26)$$

This equation forms the basis of the estimate ².

Since the true solution and therefore, true error, is unknown, we will use upper and lower bounds in terms of the *a priori* parameters. From the SVM theory, it is known that the norm of the solution $\|\mathbf{w}\|^2 < R^2$, where R is the radius of the ball centred at the origin in the feature space and which can be computed as $R \leq \max_{1 \leq i \leq \ell} (K(x_i, x_i))$ [106]. Therefore,

$$\eta < \max_{1 \leq i \leq \ell} (K(x_i, x_i))^2. \quad (4.27)$$

Now, consider the term for the norm of the error, ρ . Let \hat{y}_i be the predicted output value of y_i of the SVM model. Since the SVM for regression uses an ϵ -insensitive loss function,

$$\begin{aligned} \rho &= \frac{1}{\ell} \sum_{i=1}^{\ell} \left(y_i - \sum_{k=1}^N \beta_k(\gamma) K(x_k, x_i) \right)^2 \\ &= \frac{1}{\ell} \sum_{i=1}^{\ell} (y_i - \hat{y}_i)^2 \\ &\geq \frac{1}{\ell} \sum_{i=1}^{\ell} (|y_i - \hat{y}_i| - \epsilon)^2. \end{aligned} \quad (4.28)$$

It is clear from (4.28) that a lower bound in terms of *a priori* information should involve the number of data points, the range of the output data and the value of ϵ . Since no such bound exists in the literature, a function involving ℓ , ϵ and the range of the output data will be derived empirically.

First, let us investigate the effect of the maximum kernel values and ϵ on C . In our analysis we used several data sets that varied in the number of input dimensions, the number of data points and the range of the output data. We determined the optimal C value using the L-curve method. Numerous SVMs using various values for ϵ and three types of kernels, namely the RBF, polynomial and mixture of a RBF and a polynomial, were constructed. In Figure 4.4 we show the effect that the maximum kernel value,

²Vapnik derived in Chapter 7 of [106] a similar relation for γ as in (4.26) as part of the proof of a theorem. Vapnik used, however, a different approach. The relation $\rho_2(Af_\ell, Af) \leq 2d\sqrt{\gamma_\ell}$ can be rewritten to $\gamma_\ell \geq \rho_2^2(Af_\ell, Af)/4d^2$. A is an operator in a Hilbert space and the function ρ_2 is metric measuring the distance between the true output $Af = F$ and the predicted output Af_ℓ of the optimal solution f_ℓ . Finally, d is chosen such that $\|f\| \leq d$.

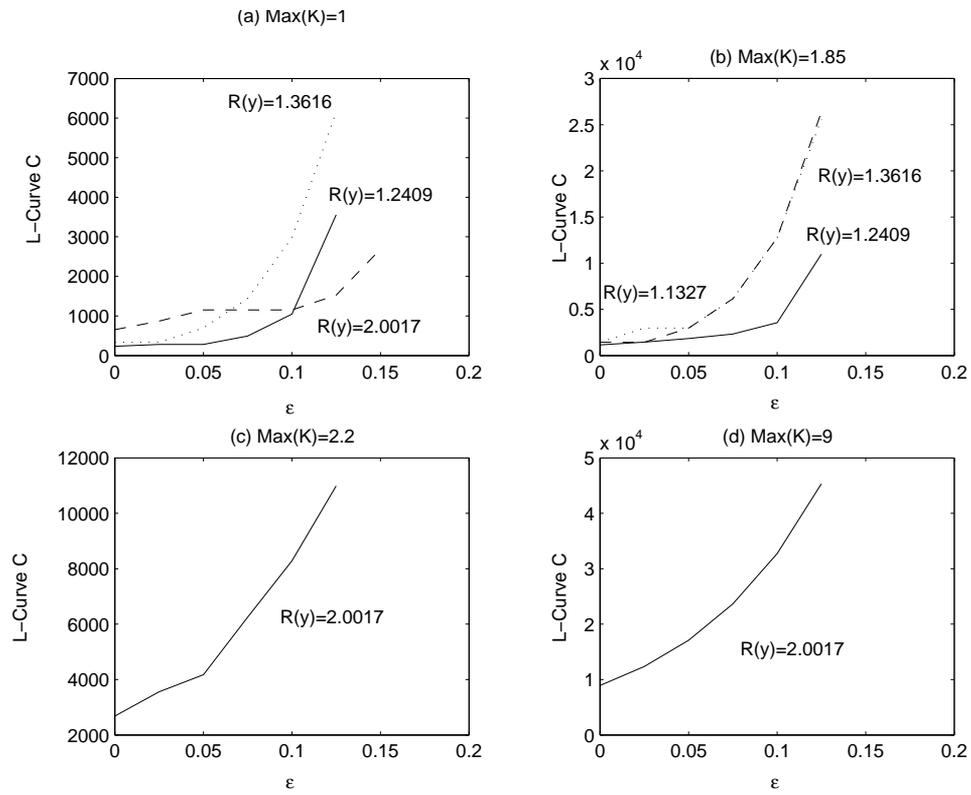


Figure 4.4: The effect of the maximum kernel value and ϵ on the optimal value of C . Increasing maximum kernel values lead to increasing C values.

an *a priori* measurable characteristic of the type of kernel, has on the value of C . It is clear that as the maximum value of the kernel increases, the magnitude of the optimal C is increased as well. This confirms the role of the kernel value in estimating C . Furthermore, in all four graphs of Figure 4.4, we see the same trend with respect to increasing values of ϵ . The effect that ϵ has on C seems to follow some kind of growth function.

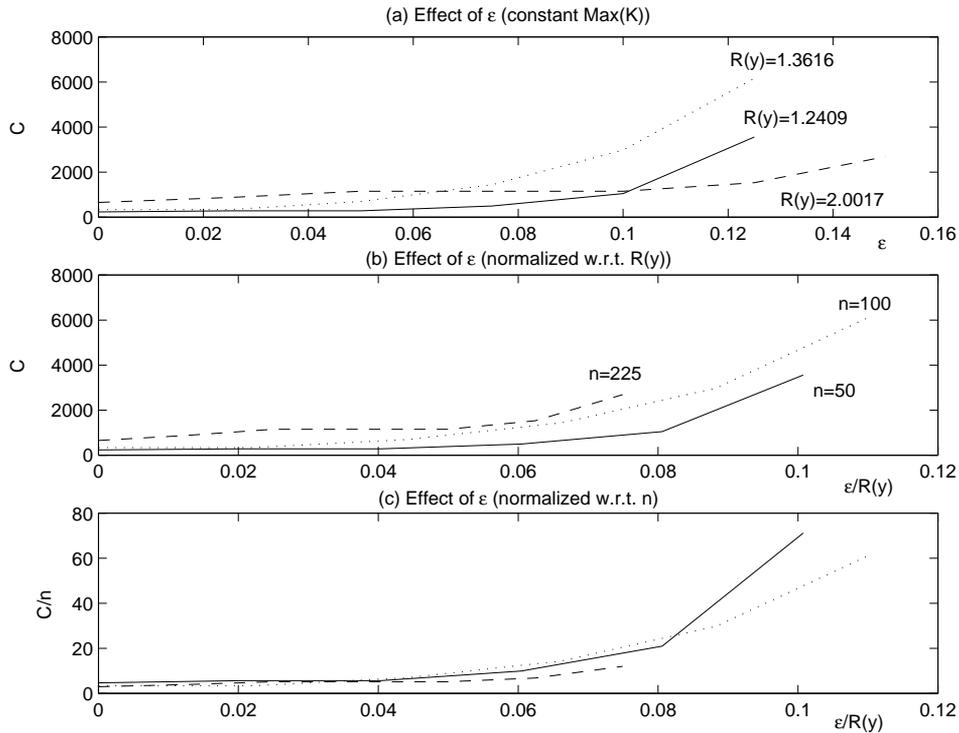


Figure 4.5: Normalisation with respect to the maximum kernel value, graph (a), the range of the output values, graph (b), and the number of data points, graph (c).

Our next step is to eliminate the influences of the maximum kernel value, the range of the output data as well as the number of data points. In Figure 4.5 the normalised results are shown. In Figure 4.5(a) the normalisation with respect to the maximum kernel value is done by considering the results for a constant maximum kernel value. Figure 4.5(b) shows the normalisation with respect to the range of the output values. The result is that the curves are in closer proximity of each other. The normalisation with respect to the number of data points is shown in Graph (c) of Figure 4.5. In this graph the curves lie almost on top of each other. All three graphs in Figure 4.5 give confirmation, at least experimentally, of the role that the maximum kernel value, the range of the output values as well as the number of data points play in the determination of C .

In Figure 4.6 the normalised data of all experiments shown in Figure 4.4 is shown. We also show a fitted growth function in terms of $\epsilon/(\text{Range}(y))$. Using the results in Figure 4.6, we arrive at the following estimation rule for C

$$C_{est} = \ell \cdot \max_{1 \leq i \leq \ell} (K(x_i, x_i))^2 \cdot 2 \exp\left(\frac{30\epsilon}{\text{Range}(y)}\right). \quad (4.29)$$

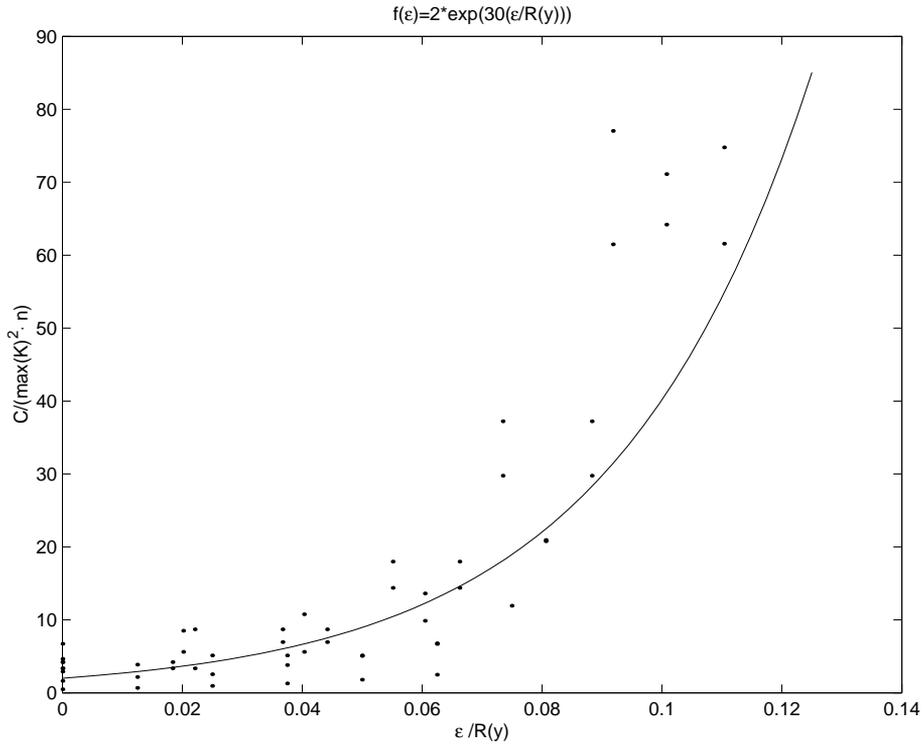


Figure 4.6: Best exponential fit of the normalised data.

4.4.3 Experimental results

In this section the estimated value of C , using (4.29) is compared to the value of C determined by using the L-curve. Several data sets with varying sizes, noise levels and dimensions were used. The results for $f(x_1, x_2) = x_1 x_2 + 1$ with $(x_1, x_2) \in [-1, 1]$ (equivalent to a continuous version of the 2D XOR problem) is presented in this section. The learning data consisted of a random sampling of this function after a noise level of $N(0, 0.05)$ was added. In Figure 4.7 the results from the L-curve approach are compared to the estimated value of C using a RBF kernel with a width of 0.2 and an ϵ of 0.05.

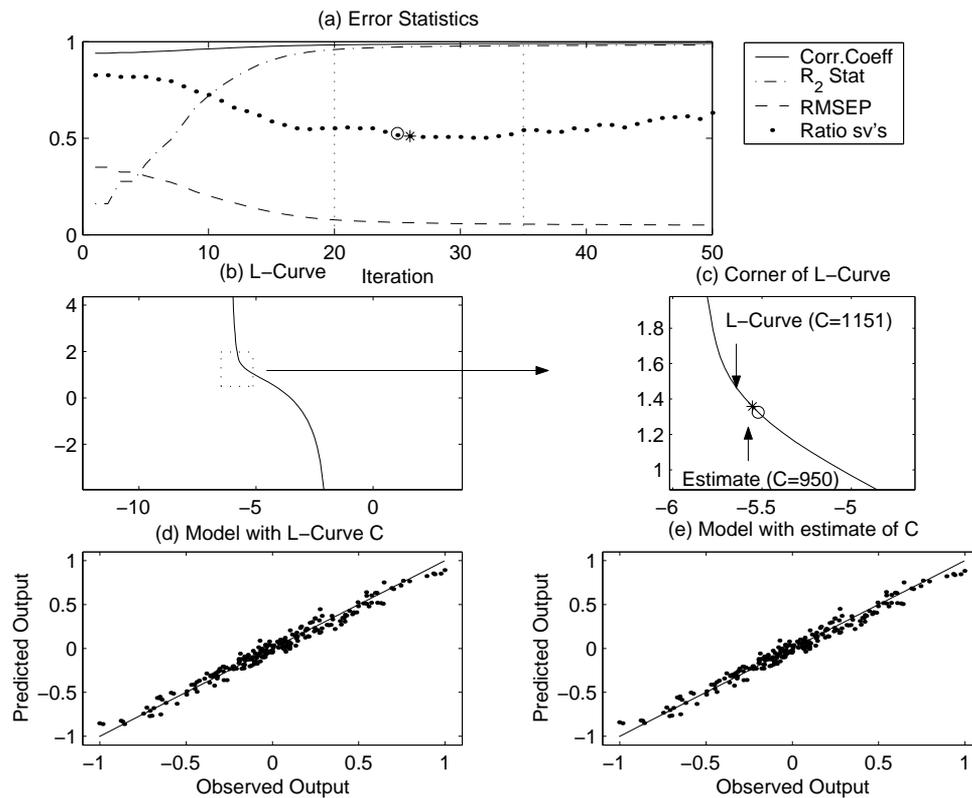


Figure 4.7: Results for a RBF kernel of width 0.2. The (near) optimal value of C is indicated by an asterisk and the estimated value of C by a circle. (a) Error statistic for each iteration step in the L-curve method. The L-curve is shown in (b) and the corner point of the L-curve in (c). In (d) and (e) the performances of the model using the optimal C and the model using the estimated C are shown respectively.

The L-curve approach requires the building of several models for increasing values of C . The range of values for C considered needs to be large enough, otherwise the corner point of the L-curve cannot be seen. Therefore, the resolution of the C -values being used, were chosen on a logarithmic scale. Figure 4.7(a) shows various error statistics of models for increasing values of C . The resulting L-curve is plotted in Figure 4.7(b). The area between the vertical dashed lines in Figure 4.7(a) corresponds to the area in the corner of the L-curve, as shown in Figure 4.7(b). The area around the corner point in the L-curve is shown more clearly in Figure 4.7(c). In Figure 4.7(a) and Figure 4.7(c) the location of the optimal C -value is indicated by the asterisk and the circle shows the location of the estimated C -value. Finally, Figure 4.7(d) and Figure 4.7(e) show the performance of the models built using the (near) optimal C and the estimated C , respectively. In this example the estimated value of $C = 950$, is rather

close to the (near) optimal value of $C = 1151$ from the L-curve, as seen in Figure 4.7(c). However, as the values of C in Figure 4.7(C) ranges from just over 200 up to 15,000, it is clear that C is a robust parameter. Therefore, the estimation needs only to predict a value of C close to the corner of the L-curve.

Table 4.1 shows the performance of SVMs using the estimation of C with respect to the performance of SVMs using the optimal C obtained from the L-curve method. More information on the R^2 and RMSEP error measures can be found in Chapter 9. It is clear from Table 4.1 that the estimated value of C produces models with error statistics that compare well with the error statistics of a model using the optimal value of C , including the percentages of support vectors.

Table 4.1: Performance of SVMs using the estimated value C and polynomial kernel of degree 2. The values in the brackets give the corresponding result when using the optimal C obtained by the L-curve method.

ϵ	Value of C	% sv's	R^2 -statistic	RMSEP
0	36000 (8929)	100 (100)	0.9704 (0.9704)	0.0657 (0.0657)
0.025	53000 (12356)	76 (74)	0.9703 (0.9702)	0.0658 (0.0659)
0.05	77000 (17100)	48 (46)	0.9699 (0.9698)	0.0662 (0.0664)
0.075	110000 (23664)	26 (25)	0.9695 (0.9692)	0.0667 (0.067)
0.1	160000 (32748)	15 (15)	0.9689 (0.9684)	0.0674 (0.0679)
0.125	240000 (45320)	8 (8)	0.9678 (0.9674)	0.0685 (0.069)

The CPU time for determining the (near) optimal value for C through the L-curve method was on average around 90 seconds. For the estimation method, the CPU time was less than 1 second. The computational advantage speaks for itself. The estimated value of C can also help to speed up the L-curve method, since one can get a good initial guess for a starting point of the algorithm.

4.5 Conclusion

Let us consider the design requirement on the robustness again.

The inferential sensor should be robust to noise and outliers in the data.

In this chapter we have shown that the SVM for regression with linear ϵ -insensitive loss function is in close spirit of robust estimators. We further discussed how the performance of the SVM is affected by the noise power with respect to the value of ϵ . It was also shown how the SVM using linear ϵ -insensitive loss is unaffected by the presence of outliers. Thus the linear ϵ -insensitive loss function makes the inferential sensor robust to noise and outliers.

The ϵ -SVM and ν -SVM using the quadratic ϵ -insensitive loss were also derived and their corresponding algorithmic pseudo codes were given.

We further presented a method for estimating the regularisation parameter C for SVM for regression. The estimation is based on results from the analysis of the L-

curve method. It was mentioned in the introduction that choosing a value for C should involve taking into account several factors, including the kernel function and the noise level. These factors are all present in the heuristic proposed.

Comparing the values of C obtained from the L-curve method with the values determined by the estimate, using several data sets, showed that the estimates of C -values are in close proximity to the optimal C . Furthermore, the difference in performance between a model using the C -value determined by the L-curve and a model using the C estimated by the method, is very small and often negligible.

The computation time needed to determine a good estimate of the optimal C is a fraction of the time needed to determine the (near) optimal value of C by means of the L-curve method. Therefore, the proposed estimation method can be used for online applications in industry. In particular, if one needs to make a quick assessment of whether a given data set can be solved with the SVM method or if a given kernel function is an appropriate choice, the fast and robust estimation method is extremely useful.

In the results given in this chapter, only the ϵ -SVM was considered with quadratic loss, assuming that the ϵ is known *a priori*. Future work needs to be done for deriving similar estimates for the ϵ -SVM with linear loss as well as the ν -SVM [10],[89], where the expected ratio of support vectors is used instead of ϵ .

Part II

Application Stability Requirements

Chapter 5

Generalisation Ability

5.1 Introduction

Although many measurements are being taken in a process, a process in a pilot plant cannot be run over the whole range of possible process conditions to obtain information over the whole input space. It is too expensive and very time consuming. The result is that the learning data often cover only a small part of the input space.

Therefore, when a process is in operation on the plant it may venture into operating regions that were unknown at the time of modelling. Most empirical models, such as NN's, do not extrapolate well [70]. Therefore in many inferential sensor applications efforts are made to restrict the inferential sensor's predictions to the known input space [70]. However, it is often expected by the process engineers that the model is able to predict unseen data within a reasonable distance from the known input space well. And for unseen data that are too far away, a "graceful degradation" of the model is preferable. That means, the model does not become instable and exhibit erratic behaviour.

The design requirement for the inferential sensor is therefore that it is able to predict unseen data in regions of low data density in the known input space as well as regions that are outside the known input space.

In Part I of the thesis, we showed that SVM has the ability to achieve a number of learning requirements. One feature of the SVM method is that it uses kernel functions to map the learning data (nonlinearly) into a higher dimensional feature space where the learning ability of the linear learning machine is increased. The type of kernel used has an influence on the learning machine's generalisation ability [86]. That is the ability to predict unseen data accurately. One way to interpret the generalisation ability is to consider the interpolation and extrapolation abilities of a model. Here we use the term interpolation to indicate when a prediction of an unseen data point within the known input space is being made, including regions where the data density is low. Extrapolation occurs when the unseen data point originates outside the known input space and the model is forced to predict in a region where nothing is known.

Every kernel has its advantages and disadvantages with respect to interpolation and extrapolation abilities. Often the kernel has either good interpolation abilities or

good extrapolation abilities. Seldom it has both. Preferably the “good” characteristics of two or more kernels should be combined. In this chapter it is shown that using mixtures of kernels can result in having both good interpolation and extrapolation abilities. The performance of this method is illustrated with an artificial as well as an industrial data set.

The layout of the chapter is as follows. In the first section the interpolation and extrapolation capabilities of SVMs using the RBF and polynomial kernels are investigated. The mixed kernel approach is introduced in section two. In the final section, the improved interpolation and extrapolation ability of the mixed kernel is shown.

5.2 Interpolation and extrapolation

Consider the SVM model in terms of the determined weights and bias, the chosen kernel function and the given support vectors,

$$f(\hat{\mathbf{x}}) = \sum_{i \in SV} \alpha_i^* K(\mathbf{x}_i, \hat{\mathbf{x}}) + b. \quad (5.1)$$

Much of the characteristics of the model in (5.1) are determined by the type of kernel function used. The quality of a model should not only be measured in terms of its ability to learn from the data but also its ability to predict unseen data. The model’s ability to predict unseen data within the known input space as well as outside the known input space is investigated in this section.

Polynomial and RBF kernels are used to show the difference in interpolation abilities of SVM models built using different types of kernels. The reason for analyzing these two types of kernels is twofold. Firstly, they can be used as representatives of a broader class of local and global kernels respectively. Secondly, these kernels have computational advantages over other kernels, since it is easier and faster to compute the kernel values.

For the analysis, the following sinc function with an added linear component, is used:

$$y = \frac{\sin(50x)}{50x} + x + 1, \quad (5.2)$$

where x (250 data points) was drawn from a uniform random distribution between $[-1, 1]$. See Figure 5.1. As a learning set $x \in [-0.5, 0.5]$ and the corresponding output y is used. For the SVM models the following parameters were used: $\epsilon = 0.01$, $C = 1000$ and the linear ϵ -insensitive loss-function.

In Figure 5.2 SVMs using polynomial kernels of various degrees were determined. The SVMs were trained on the data between the vertical dashed lines. Beyond these lines, the SVM will have no information available and any prediction will be the result of extrapolation. In Figure 5.2(a) it is observed that for lower degrees of polynomial kernels the extrapolation ability gets better. However, for good interpolation ability higher degree polynomials are required, as seen in Figure 5.2(b). No single choice of kernel parameter, the degree of the polynomial, results in a SVM that will provide both good interpolation and extrapolation properties.

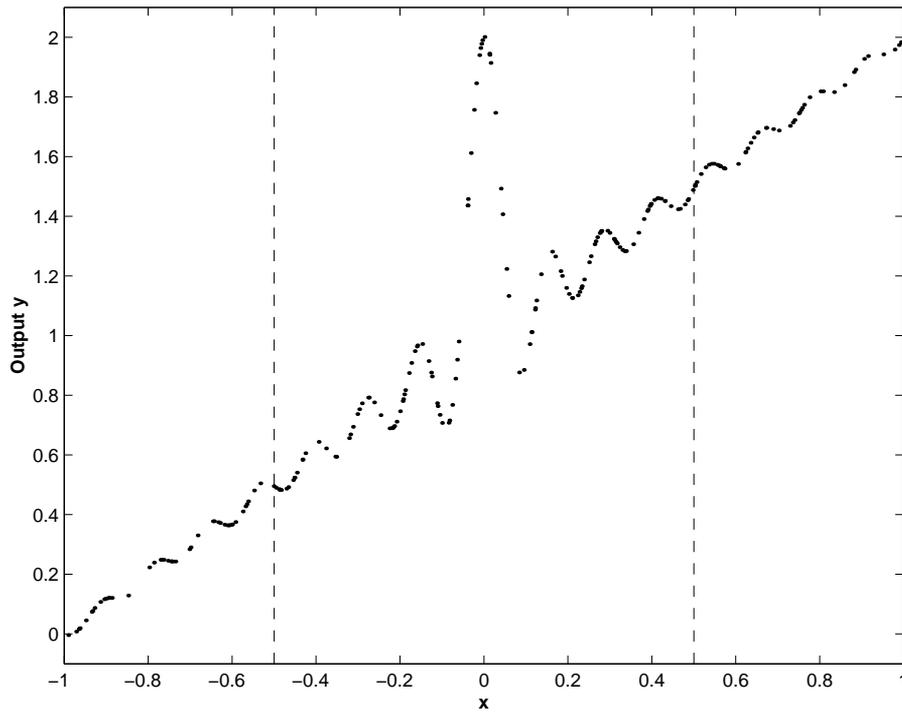


Figure 5.1: Sinc Function with added linear component as defined in (5.2). The learning range is $[-0.5, 0.5]$.

In Figure 5.3 the RBF kernel is analyzed in the same manner. The width of the kernels were varied between 0.01 and 0.25. The SVMs were again trained on the data between the vertical dashed lines. Beyond these lines, the SVM will have no information available and will need to extrapolate. When one attempts to extrapolate outside the input data range combined with the width of the kernel, there is no local information available and the prediction values level off. This is clearly seen in Figure 5.3(a). If one uses too large values of σ , as seen in Figure 5.3(b), the interpolation ability of RBF kernels decreases. Therefore, no single value of the kernel parameter, σ , will provide a model with both good interpolation and extrapolation properties.

5.3 Mixed kernel approach

From the previous section, it is observed that a polynomial kernel (a global kernel) shows better extrapolation abilities at lower orders, but requires higher orders for good interpolation. On the other hand, the RBF kernel (a local kernel) has good interpolation abilities, but fails to provide longer range extrapolation. Preferably one

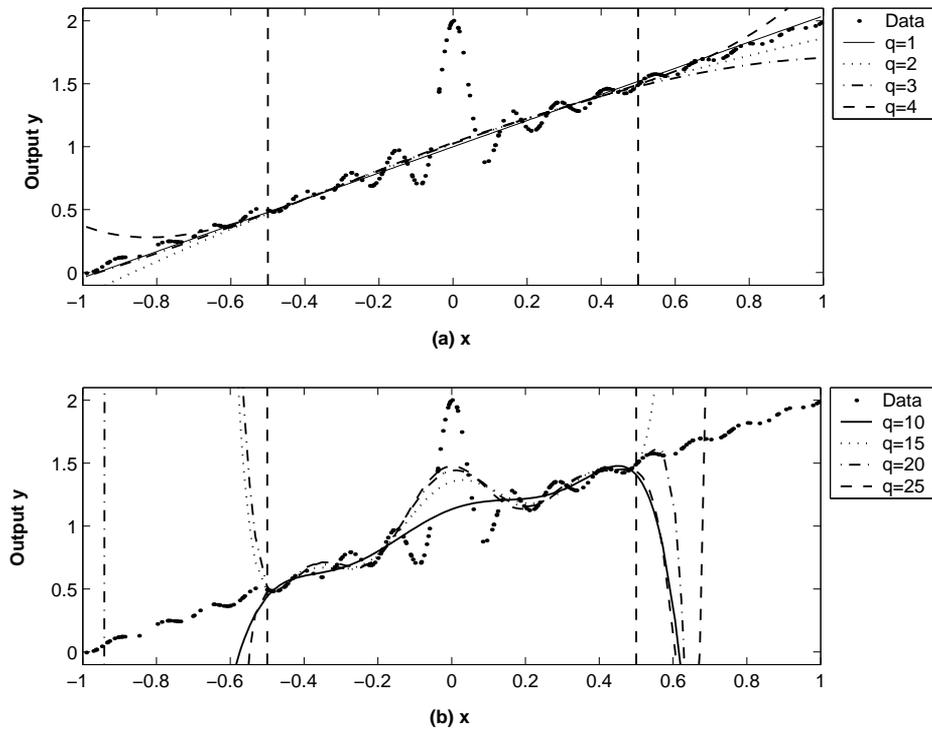


Figure 5.2: SVM using polynomial kernels (a) for degrees $q = \{1, 2, 3, 4\}$ (b) for degrees $q = \{10, 15, 20, 25\}$.

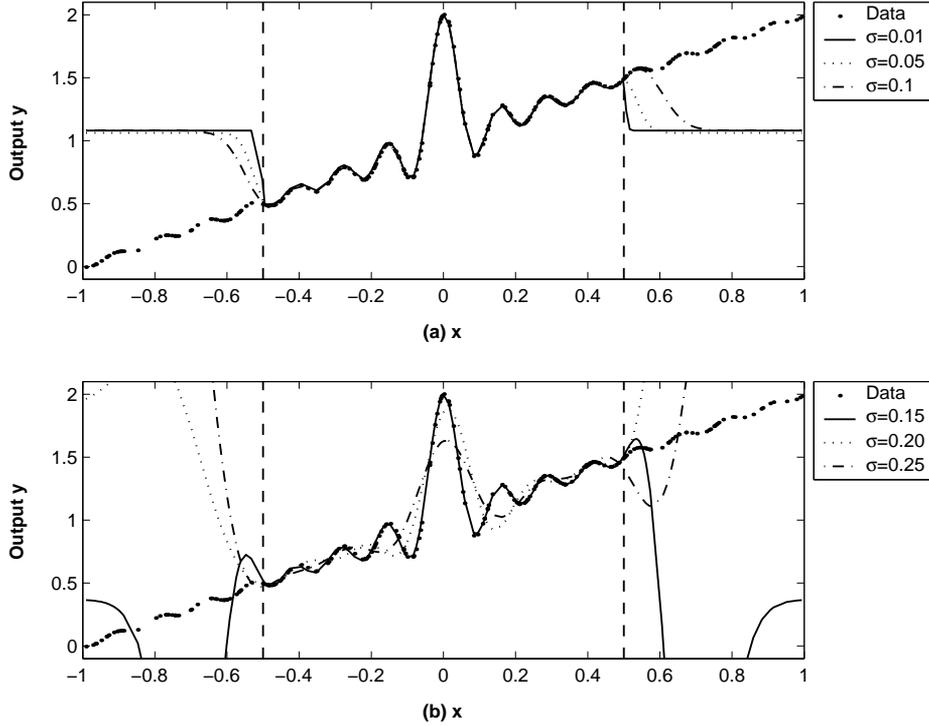


Figure 5.3: SVM using RBF kernels (a) for widths of $\sigma = \{0.01, 0.05, 0.1\}$ (b) for widths of $\sigma = \{0.15, 0.2, 0.25\}$.

wants to combine the “good” characteristics of the two kernels. Therefore, we will investigate whether the advantages of polynomial and RBF kernels can be combined by using mixtures.

There are several ways of mixing kernels. What is important though, is that the resulting kernel must be an admissible kernel [5]. One way to guarantee that the mixed kernel is admissible, is to use a convex combination of the two kernels K_{poly} and K_{rbf} , for example

$$K_{mix} = \rho K_{poly} + (1 - \rho) K_{rbf}, \quad (5.3)$$

where the optimal mixing coefficient ρ has to be determined. The value of ρ is a constant scalar.

In Figure 5.4, the effect of mixing a polynomial kernel with a RBF kernel is shown. The same example as in the cases of polynomial and RBF kernels was used to show the combined effect of using a mixture of a local and global kernels. The degree of polynomial and width of RBF were fixed to 1 and 0.15, respectively. Only the mixing coefficient was varied between 0.5 and 0.95. Smaller mixing coefficients are not shown, because the effect of the global kernel only then becomes significant. Figure 5.4 shows that the kernel has not only a local effect, but also a global effect. By increasing the

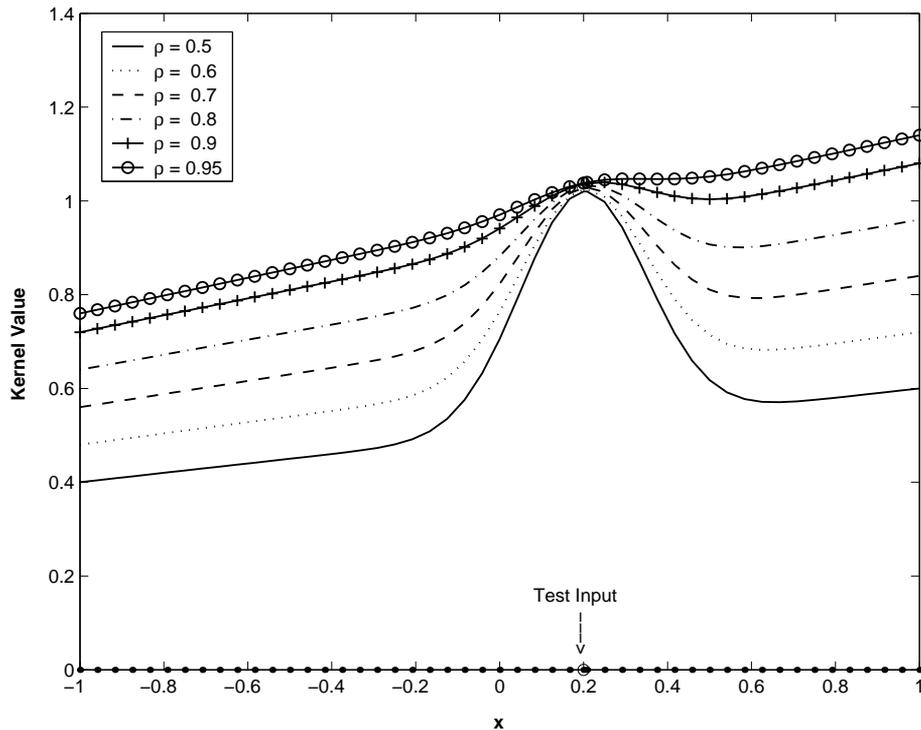


Figure 5.4: Example of a mixed kernel with first degree polynomial and RBF with width 0.15.

mixing coefficient, the influence of the global kernel is also increased. Similar to the 3D-view of the RBF and polynomial kernels shown in Chapter 3, the 3D-view of the mixed kernel in Figure 5.5 give an interesting perspective on the combined effect of the two kernels.

Another possibility of mixing kernels is to use different values of ρ for different regions of the input space. ρ is then a vector. Through this approach the relative contribution of both kernels to the model can be varied over the input space. In this section, a uniform ρ over the entire input space is used.

The same data set used in the previous section will also be used to examine a mixture of polynomial and RBF kernels. In Figure 5.6 the SVM predictions of the test set are given using the mixing coefficients ρ in the range $[0.5, 0.99]$. Here, a value of $\rho = 0.95$, for example, means that the relative contribution of the polynomial kernel to the mixed kernel is 95% whilst the RBF kernel contributes the remaining 5%. Again, the SVMs were trained on only the data between the vertical dashed lines. Beyond these lines, the SVM will have no information available and any prediction is the result of extrapolation. Note that the SVM models using the mixed kernel have far better extrapolation capabilities than the SVMs using either the polynomial or

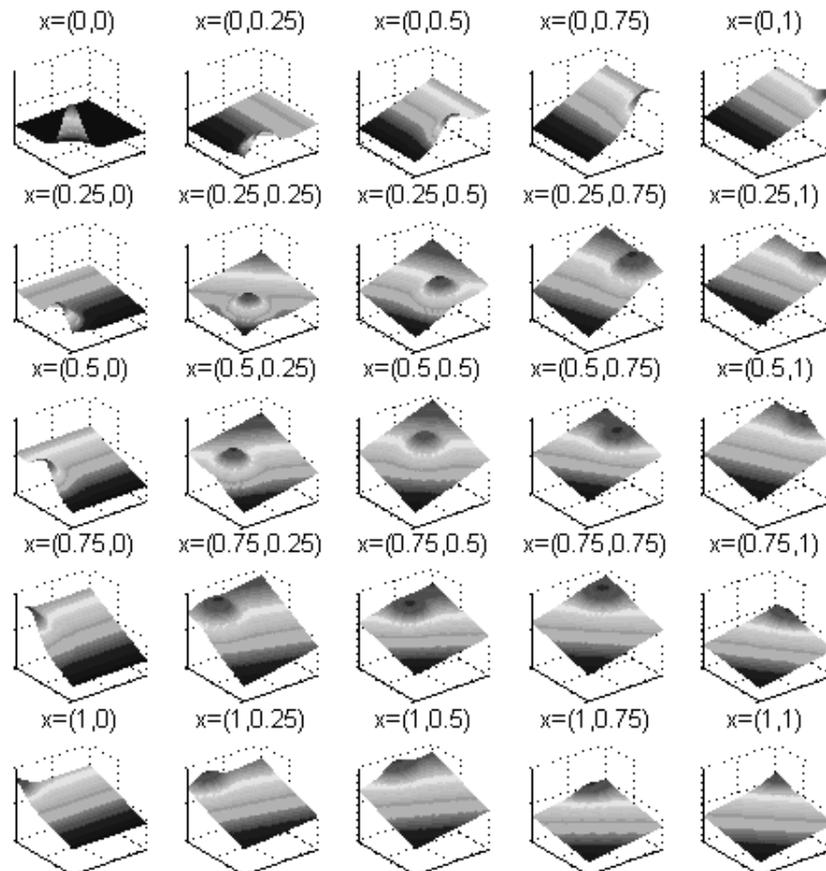


Figure 5.5: A 3D-view of the effect of the mixing the RBF and polynomial kernels.

RBF kernels on their own.

What is remarkable, is that only a “pinch” of a RBF kernel ($1 - \rho = 0.01$) needs to be added to the polynomial kernel to obtain a combination of good interpolation and extrapolation abilities for the same kernel parameter values. It is striking that both these properties can be achieved with a single choice of parameters for a mixed kernel. Using higher degrees of polynomials or larger widths of RBF kernels did not produce better results. Using the same criterion as in the previous section, the optimal value of ρ was found to be 0.95.

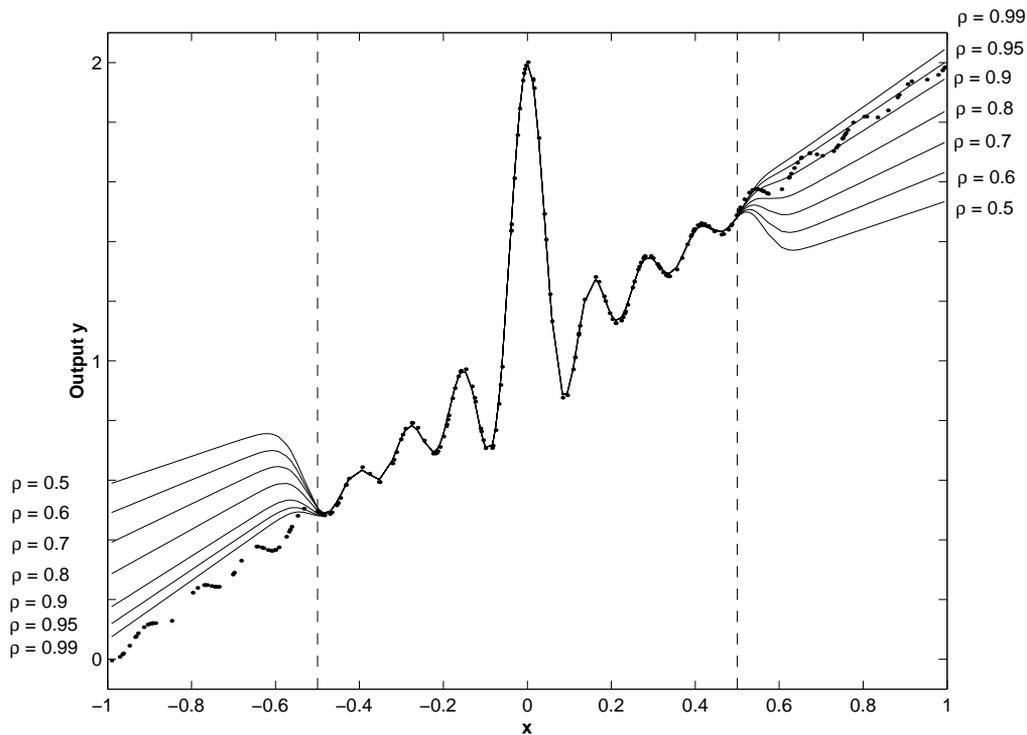


Figure 5.6: SVM using mixed kernels of first degree polynomial, $\sigma = 0.05$ width of RBF and $\rho = \{0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99\}$.

5.4 Industrial application

An industrial data set was used to test whether the advantages of using mixed kernels also apply to noisy, real-life data. The data were obtained from a process of The Dow Chemical Company in Freeport, USA, from 1997 to 1999. Samples were taken approximately every six to eight hours and sent to a lab for analysis. The input variable, a ratio of two temperatures, was selected after an extensive feature selection process using NNs. All observations with temperature ratio smaller than 0.9 and larger than 1.1 together with a number of randomly selected observations within the range were used as test data. The rest were used as learning data. The learning set consisted of 627 observations and the test set had 561 data points. The learning input data were range-scaled to $[0, 1]$ and the resulting scaling parameters were then used to scale the test data.

SVM models using a polynomial kernel and RBF kernel individually were then determined. In all models an ϵ -value of 5 was used, because that is an acceptable error level in the process. For the SVM Models with polynomial kernels, degrees from 1 up to 5 were used. The predictions of the learning set for the different models are

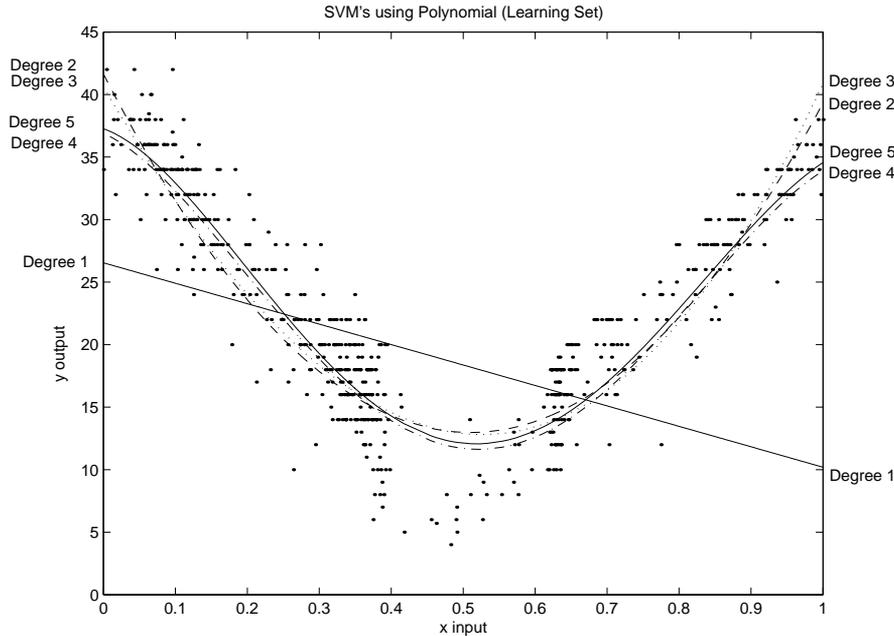


Figure 5.7: SVM predictions of the learning set, using polynomial kernels of various degrees.

shown in Figure 5.7 and the predictions for the test set can be found in Figure 5.8. Figure 5.8 also indicates where a model interpolates and where it extrapolates. The model interpolates where data has been available during the training phase, indicated by the area between the vertical dashed lines. Outside this area, the model will extrapolate since there was no information available during the learning process.

The number of support vectors used by each model ranges between 8 and 11 percent of the learning data. Note that the SVMs using the second and third degree polynomials are able to roughly follow the trend of the test data, but fail to predict it accurately. Increasing the polynomial order does not improve the generalisation capabilities.

In the SVM Model using RBF kernels we used widths ranging between 0.1 and 0.5. Again in all models an ϵ -value of 5 was used.

The predictions of the learning set are shown in Figure 5.9. Figure 5.10 displays the predictions of the test set as well as the known input space (the area between the vertical dashed lines). The number of support vectors used by each model ranges between 7 and 10 percent of the learning data. One inherent property of RBF kernels is clearly seen in Figure 5.10: although the RBF kernel interpolates very well within the known input space, it fails to extrapolate outside the range of its width.

In both cases the SVM used more or less 10% of the learning data as support vectors. Both models predict the learning set fairly well, but fail to predict the test

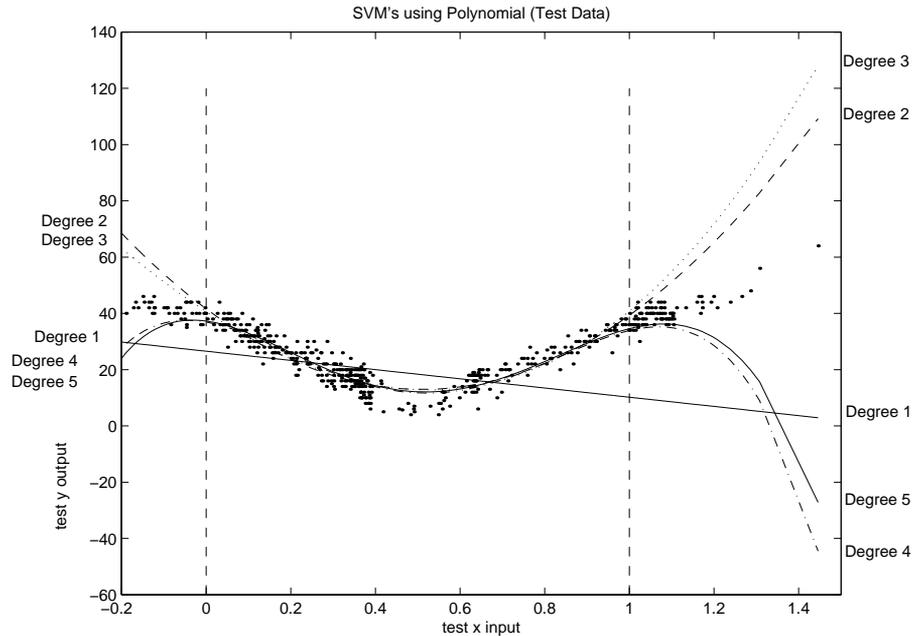


Figure 5.8: SVM predictions of the test set, using polynomial kernels of various degrees.

set outside the known range accurately. Increasing the number of support vectors will not necessarily improve the predictive power of the model, since there is a threshold of the number of support vectors with respect to generalisation ability. If too many support vectors are used, the model is overfitting and models noise as well. The resulting predictions of the test set will actually become worse. Furthermore, the intrinsic complexity of the data is fairly low and does not need the use of more support vectors. In fact, the number of support vectors can be decreased to less than 5%, but at the expense of accurately predicting the sharp turning point in the data around the ratio of 0.5.

In Figure 5.11, the mixed kernel approach is shown. From the analysis of the kernel parameters it is found that an appropriate choice for kernel parameters is a second degree polynomial, combined with a RBF of width 0.15 and mixing coefficient of 0.98. In Figure 5.11, the model is displayed. The top graphs shows the prediction of the learning set well as the ϵ -insensitive tube and support vectors (encircled points). The bottom graph depicts the prediction of the test set. The number of support vectors used is also in the region of 8% of the learning data. The model of the mixed kernel is able to interpolate the sharp turning point of the learning data as well as extrapolate outside the known input space.

Although this example is only one-dimensional, the mixed kernel approach was also applied to higher dimensional problems and similar generalisation performance

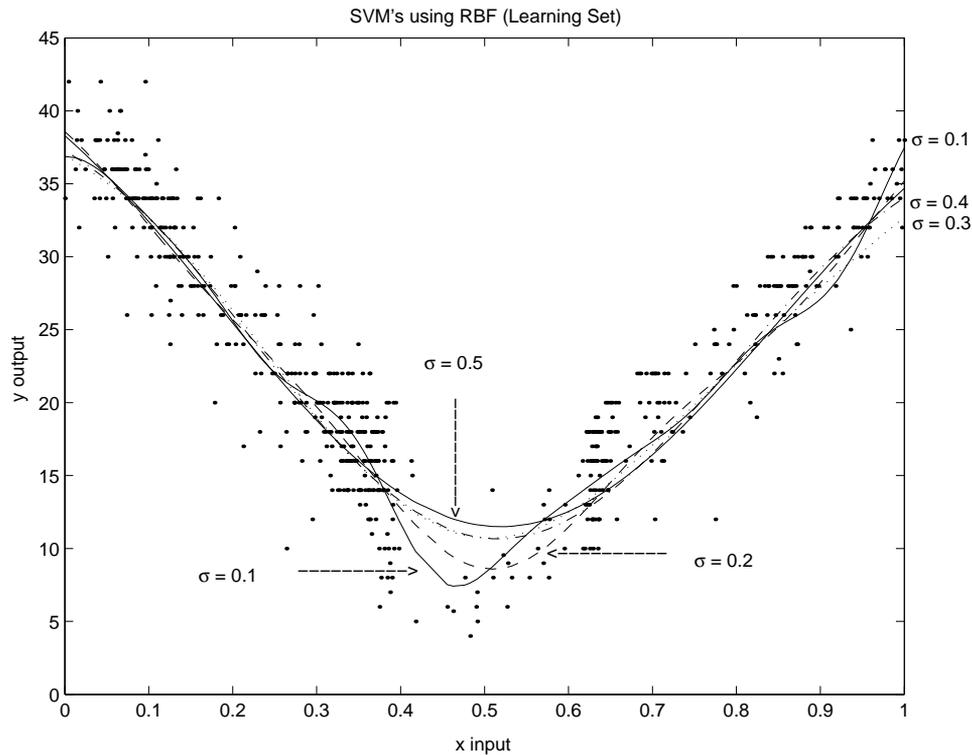


Figure 5.9: SVM prediction of Learning set, using RBF kernels of various widths.

improvements were found.

5.5 Conclusion

Let us state the original design requirement again:

The inferential sensor is required to predict unseen data in regions of low data density in the known input space as well as regions that are outside the known input space.

It was shown in this chapter that, where the RBF kernels fail to extrapolate and a very high degree polynomial kernel is needed to interpolate well, the mixture of the two kernels is able to do both. Furthermore, a model that interpolates and extrapolates well can be built using a single choice for each of the kernel parameters.

Having the ability to both interpolate and extrapolate well, now opens the door for making use of prior information, which is one of the design requirements. If, for example, the asymptotic behaviour of a process is known from fundamental models, this information can be used as prior information. The SVM using a mixture of

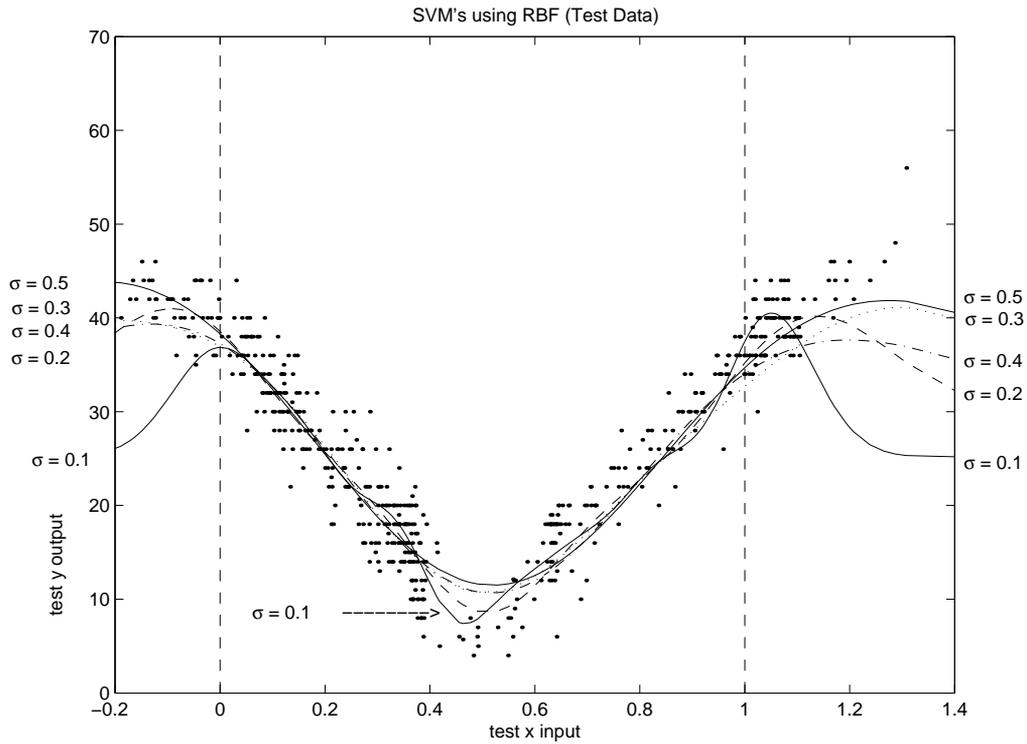


Figure 5.10: SVM predictions of the test set, using RBF kernel of various widths.

kernels then will not only be able to learn from the data but also take into account the behaviour of a process in the limit. This design issue will be address in Chapter 7.

Further investigation needs to be done into why only a very small percentage of the RBF kernel is needed. Other interesting questions remain also, like how could the performance be improved by using local density related values for the mixing coefficient in different regions of the input space. There are of course other kernels that could be used for mixing. For example, mixing several RBF kernels may be useful for problems with a nonuniform data density in the input space [111].

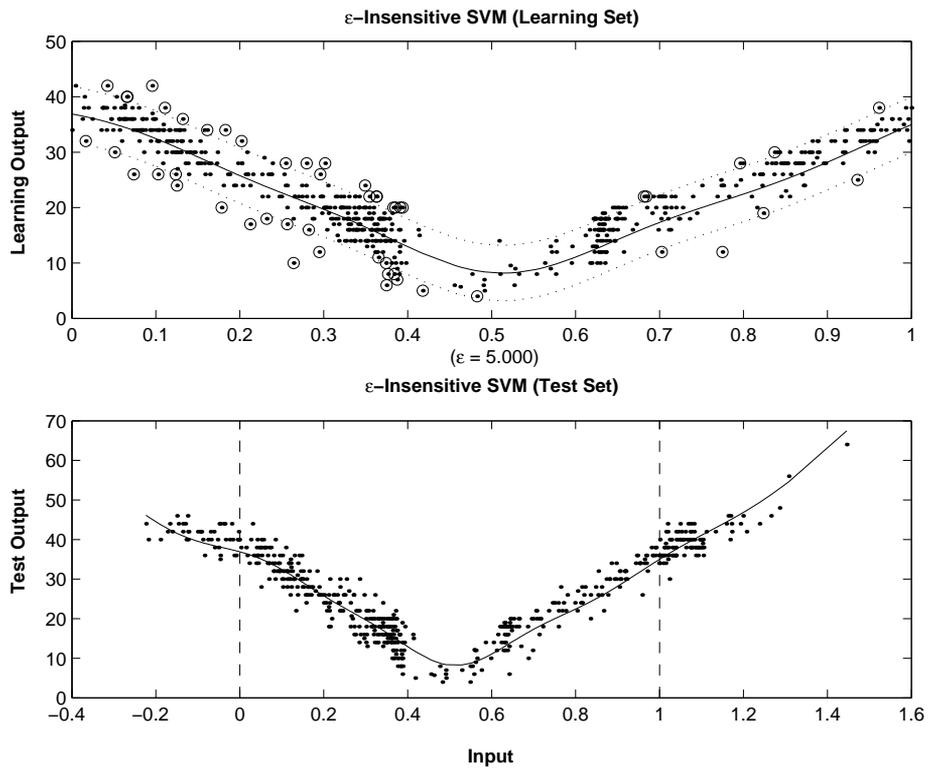


Figure 5.11: SVM using mixture of polynomial kernel of degree two, an RBF kernels of width 0.15 and a mixing coefficient of 0.98.

Chapter 6

Data Compression and Outlier Detection

6.1 Introduction

The quantity of data being generated in chemical plants increased tremendously in the last decade. In contrast to fifteen years ago when there were never enough data, the size of data sets today are becoming unmanageable. In fact, the following has been said [13].

...companies today are manipulating data in the terabyte (one trillion bytes) to pedabyte (one thousand terabytes) range. If bytes were raindrops, that would be enough to float the QEII.

One could argue that in time the increasing speed and memory capabilities of computers would solve the problems involving data management. However, the amount of data gathered increases probably at the same rate as the speed and memory of computers, maybe even faster. Therefore, the problem will remain. Compressing data sets without loss of information is an essential capability for modern modelling tools [14]. Reducing the size of a data set requires the removal of redundant data. That is remove duplicate data or data that does not add any new information to the data set [14].

Another issue that also involves the removal of data points is the detection of outliers. Outliers are unusual data points that are not consistent with most of the data points [11]. Since in the design requirement on robustness the inferential sensors were made insensitive to outliers, it would appear that outlier detection is not required anymore. However, outliers often contain useful information on abnormal behaviour of the process described by the data [1]. Consequently, outlier detection still needs to be done in order to fully understand the behaviour of the processes under consideration.

Many data compression and outlier detection methods exist. However, there is an increasing awareness that most of these approaches are distance- or density-based which make them inappropriate for high-dimensional data [1],[98]. The reason for this

has been linked to the curse of dimensionality and in particular with Property 4 in Chapter 3. Recall that Property 4 states that in high-dimensional spaces almost every point is considered to be an outlier. Therefore in recent years scientists expressed the need for model-based approaches [85],[48].

It has been observed in the SVM literature that the SVM may be used for detecting outliers [11]. The SVM's robustness with respect to outliers, as shown in Chapter 4 and the fact that the outliers are part of the support vector set makes it potentially suitable to use as a model-based outlier detection method. In [99] it was shown how SVMs for classification can be used to detect outliers. To our knowledge there has not been extensive research in applying SVM for regression in outlier detection applications.

Another observation made by a number of researchers is that the SVM's ability to represent the model in terms of the support vectors could be used for data compression purposes [87],[10],[106],[107]. The use of SVMs for classification for data compression was discussed in [76]. The potential of using SVMs for data compression in inferential sensor applications has also been noted in [22]. However, little research has been done to investigate these claims for the regression case.

In this chapter of the design thesis we explore the use of SVMs for performing outlier and redundancy detection tasks. In the first section we present an approach to perform outlier detection. Inverting the reasoning presented in the first section, the approach can be used for redundancy detection purposes, which is discussed in the second section.

6.2 Outlier detection

Consider the dual form linear ϵ -SVM from Chapter 2,

$$\begin{aligned} \text{maximise} \quad & \sum_{i=1}^{\ell} y_i (\alpha_i^* - \alpha_i) - \epsilon \sum_{i=1}^{\ell} (\alpha_i^* + \alpha_i) \\ & - \frac{1}{2} \sum_{i,j=1}^{\ell} (\alpha_i^* - \alpha_i) (\alpha_j^* - \alpha_j) K(\mathbf{x}_i, \mathbf{x}_j), \end{aligned} \quad (6.1a)$$

$$\text{subject to} \quad \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) = 0, \quad (6.1b)$$

$$0 \leq \alpha_i, \alpha_i^* \leq \frac{C}{\ell}, \quad i = 1, \dots, \ell. \quad (6.1c)$$

In Chapter 4 it was shown that the linear ϵ -SVM, i.e. the ϵ -SVM using the linear ϵ -insensitive loss function, remains virtually unaffected by the presence of outliers even though the outliers are part of the set of support vectors. The reason for this is that the Lagrange multipliers obtained by solving (6.1) are upper bounded by constraint (6.1c). In fact, all support vectors that are not strictly on the ϵ -insensitive zone, including the outliers, will have an absolute value equal to the upper bound. Therefore, if outlier detection was to be performed by the SVM one would have to

inspect the values of the Lagrange multipliers and in particular those that hit the upper bound in (6.1c).

Consider the constraints on the Lagrange multipliers in (6.1c). The Lagrange multipliers α_i and α_i^* are bounded from above by C/ℓ , where C is the regularisation parameter. In optimisation theory it is a well-known fact that a positive Lagrange multiplier indicates that the corresponding constraint in the primal formulation of a QP problem is active at the optimal solution [17],[51], [6]. If the Lagrange multiplier's value hits the upper boundary, it means that in the case of SVMs the observed data point lies outside the ϵ -insensitive zone and consequently has a positive slack variable. If the corresponding slack variable ξ (or ξ^*) has a large value the data point in question is far outside the ϵ -insensitive zone. Therefore, Lagrange multipliers with upper bound values as well as large slack variable values indicate unusual data points and can be considered as possible outliers.

Of course, based on a single model, one cannot conclude that a data point is a possible outlier for one would then rely heavily on the quality of a particular model. Several models of varying complexity should be constructed. For each model a data point is identified as a suspected outlier if it has a Lagrange multiplier value close to the upper boundary and its corresponding slack variable is large. We consider the slack variable to be large if it is larger than 0.5 times the standard deviation of the predicted output values. Next we determine the number of times that a data point is suspected to be an outlier and plot the frequency of suspicion at increasing rates.

Consider for example the data set in Figure 6.1. The data set is the same one used to illustrate the robustness of the SVM in Figure 4.1 of Chapter 4. Given the data in Figure 6.1, an RBF kernel with $\sigma = 0.2$ and $C = 500$, the approach described above was followed and in Figure 6.2 the obtained frequency information is plotted. Figure 6.2(a) shows per iteration which data points are suspected to be outliers. In Figure 6.2(b) these frequency rates are plotted in increasing order. Note the sharp increase in the frequency at the tail of the detection rate. In our research we observed that this is characteristic for data sets where there are outliers present. The predictions of the corresponding SVMs as well as the detected outlier are shown in Figure 6.3. The various SVMs behave differently in the proximity of the outlier. After removing the detected outlier, the procedure is repeated until no outliers are detected anymore or the various SVM have the same predictive capabilities. This predictive capabilities is measured using for example the RMSEP error measure. If the standard deviation in the RMSEP values of an iteration is small, the various SVMs have equal predictive capability indicating that no large errors are present anymore. In Figure 6.4 the frequency plots after five detection iterations are shown. Note that the curve in Figure 6.4(b) is almost a straight line. There is no sharp increase in frequency at the tail of the detection rate. The increase is therefore only due to the shrinking of the ϵ -insensitive zone. In Figure 6.5, it is clear that the SVMs have the same behaviour and the standard deviation of the RMSEP values is therefore quite small. The outliers detected following the procedure described above can be seen in Figure 6.6.

It is assumed that the data set has a reasonable noise level so that the true outliers are not masked by noise. Furthermore, one should remember that as outliers are unusual data points it is expected that there are only a few outliers present.

Next we give the algorithmic pseudocode of the outlier detection procedure dis-

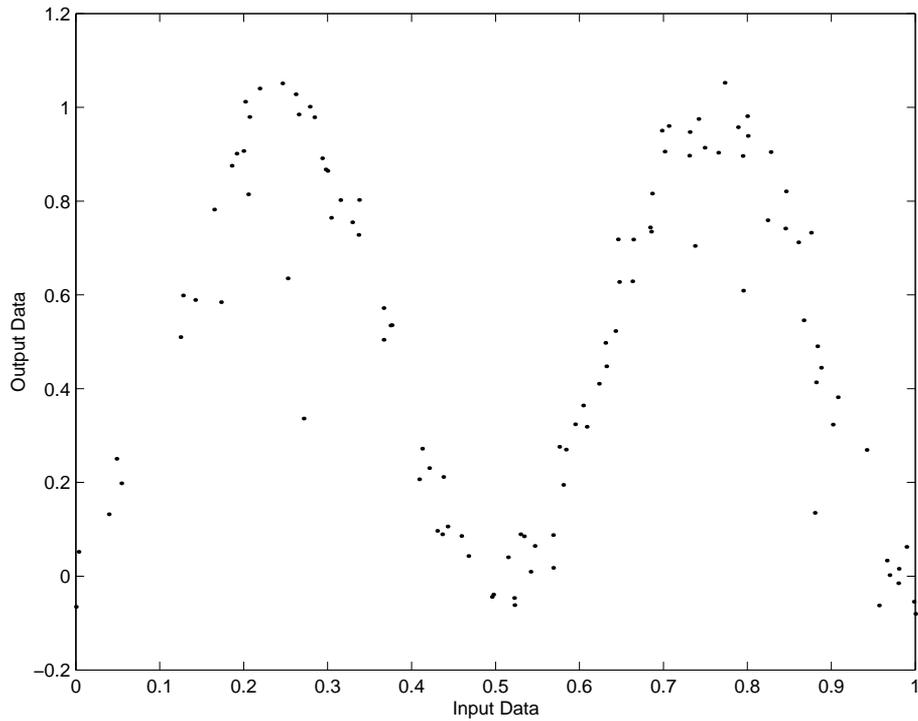


Figure 6.1: Example 1 of Chapter 2 with added outliers.

cussed and illustrated above in Algorithm H.

Algorithm H. Outlier Detection

-
- Step 1:** Select the learning data $(\mathbf{x}_i, y_i), i = 1, \dots, \ell$, kernel function K and C .
- Step 2:** Determine maximum ϵ for which support vectors exist:
- set $\nu = 0.01$.
 - run Algorithm C
 - set $sv_num = length(SV)$
 - set $\epsilon_max = \epsilon$
 - repeat until $sv_num = 0$,
 - set $\nu = \nu + 0.05$
 - run Algorithm C
 - set $sv_num = length(SV)$
 - set $\epsilon_max = \epsilon$ - end
- Step 3:** Set $it_num = 20$.

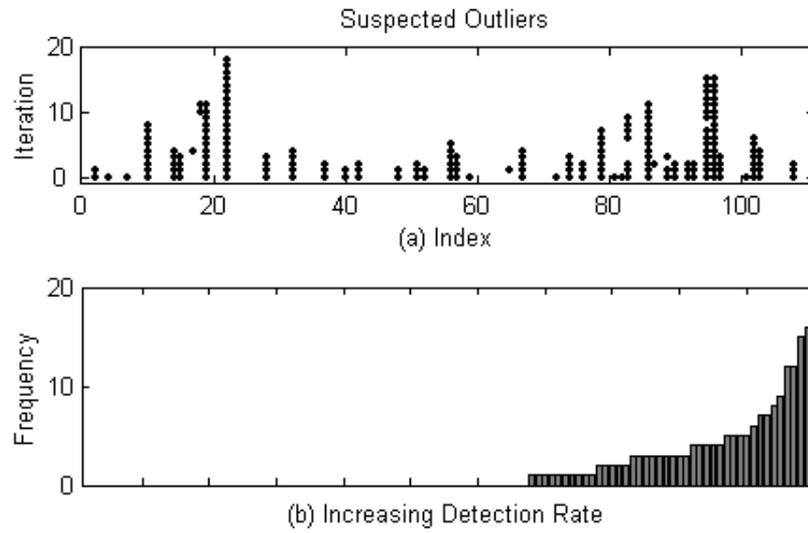


Figure 6.2: The frequency of suspected outliers.

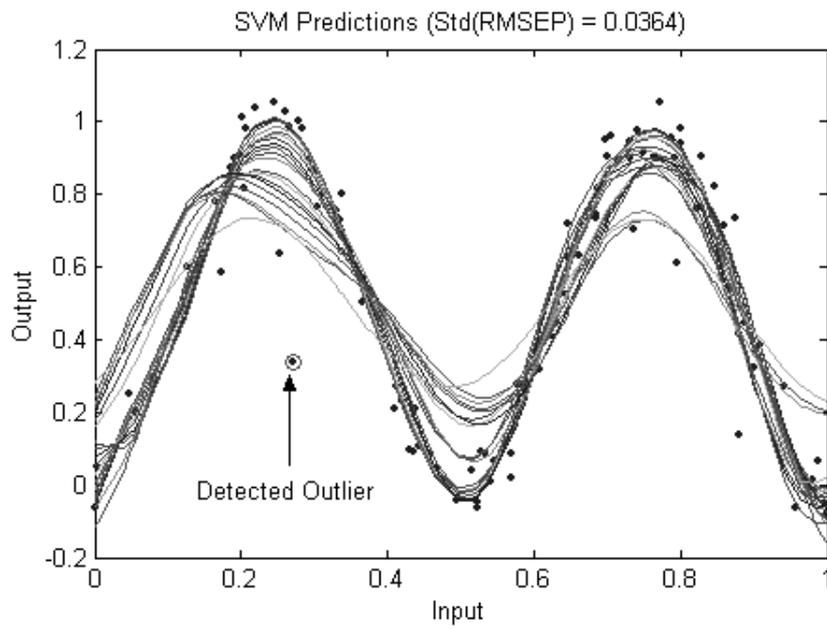


Figure 6.3: Predictions of various SVM models and detected outlier.

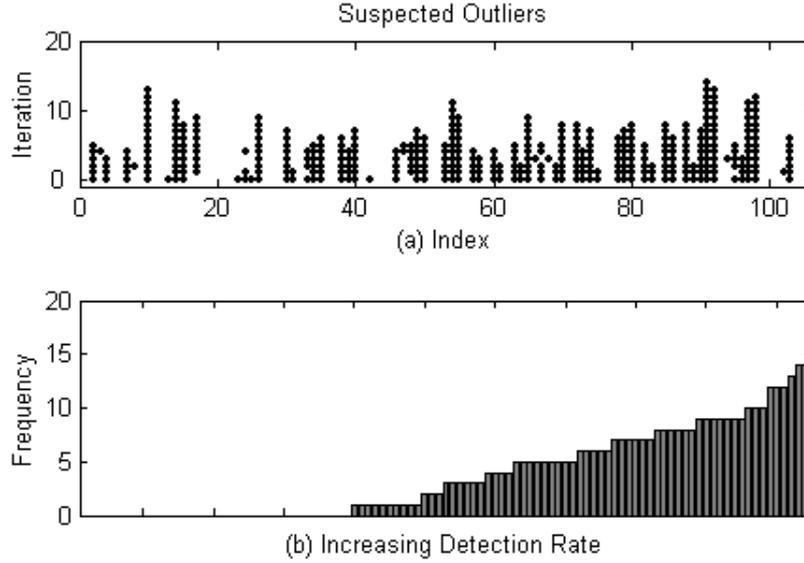


Figure 6.4: The frequency of suspected outliers after five iterations.

- Step 4:** Construct vector ϵ consisting of it_num values linearly distributed from 0 to ϵ_max .
- Step 5:** Set outlier index matrix Out_I as an $\ell \times it_num$ matrix of zeros.
- Step 6:** For $k = 1$ to it_num
run Algorithm B using $\epsilon(k)$
identify possible outliers indexes as:

$$I = \{i | \mathbf{w}_i = \frac{\epsilon}{\ell}, i = 1, \dots, \ell\}$$
for $i = 1$ to ℓ
calculate $\hat{y}(i) = f_{\omega, b}(\mathbf{x}_i)$.
calculate the errors as: $\xi(i) = y(i) - \hat{y}(i)$
end
calculate the standard deviation σ of ξ 's
identify indexes of large errors as:

$$L = \{i | \xi(i) > \frac{1}{2}\sigma, i \in I\}$$
set suspected outlier indexes set as: $S = I \cap L$
set $Out_I(S, k) = 1$
calculate the RMSEP value for the predictions:

$$R(k) = RMSEP(y, \hat{y})$$
end
- Step 7:** Calculate frequency of indexes in Out_I :

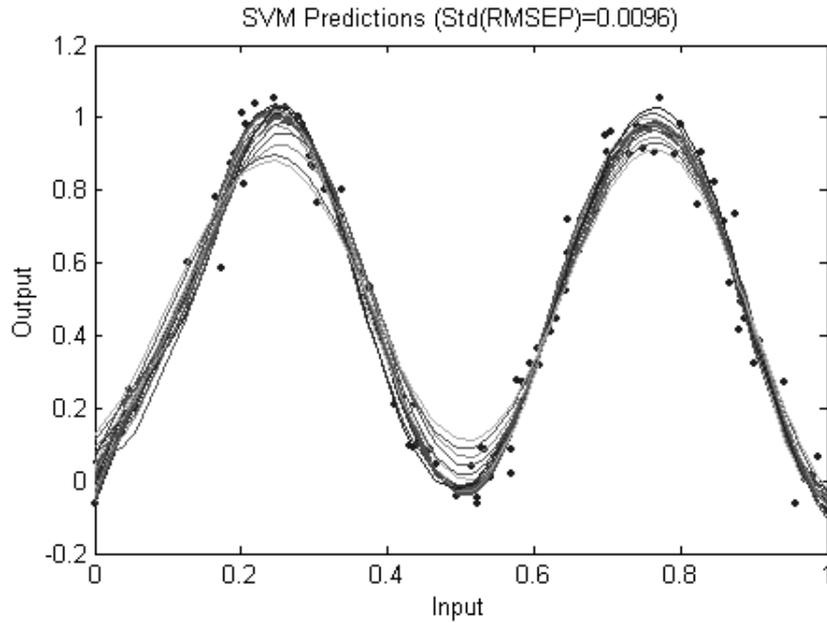


Figure 6.5: Predictions of various SVM models after five iterations.

```

for  $i = 1$  to  $\ell$ 
     $Freq(i) = \sum_{j=1}^k Out_I(i, j)$ 
end

```

Step 8: Calculate the largest frequency: $M_Freq = \max_i Freq$.

Step 9: Determine the standard deviation of the RMSEP values: $S = std(R)$.

Step 10: If $M_Freq > 15$ and $S \leq 0.01$ then
 $outlier = \{i | Freq(i) = M_Freq, i = 1, \dots, \ell\}$
 remove $\mathbf{x}(i)$ and $y(i)$ from data set
 return to step 2 using the reduced data set
 else
 stop
 end

6.3 Data reduction

For redundancy detection, we invert the reasoning followed in the outlier detection approach. Now we make use of the fact that those vectors that contain information

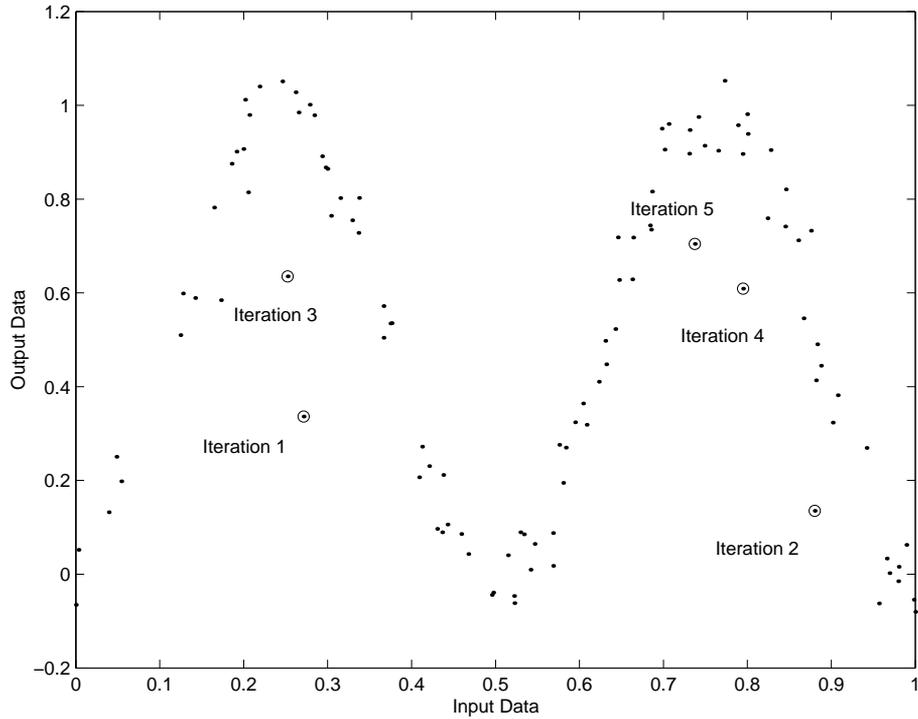


Figure 6.6: Detection of outliers in five iterations.

are identified by the SVM method as support vectors. Therefore, in order to detect data points that are redundant, one has to inspect the non-support vector data points, i.e. the data points with zero weights.

Consider the last constraint (6.1c) in the dual formulation of the SVM problem in (6.1). The Lagrange multipliers α_i and α_i^* are bounded from below by 0. Recall that the Lagrange multipliers indicate whether the corresponding constraints of the primal formulation of a QP Problem are active or inactive at the optimal solution. A zero Lagrange multiplier value indicates that the constraint is not active at the optimal solution. Note that each data point has two constraints in the primal formulation: One for when the data point lies above the approximating function and one for when the data point lies below the approximating function. Thus there are two Lagrange multipliers, α_i and α_i^* , associated with data point i . However, both the constraints cannot be active at the same time since a data point cannot lie above and below the approximating function at the same time. Therefore both the Lagrange multipliers cannot be non-zero at the same time either. A non-support vector data point is therefore a data point for which both Lagrange multipliers are zero at the same time. Such a data point may be considered as a possible redundancy, since according to the model it does not contain information necessary for the model to be used when predicting an unseen data point.

Again, one should not conclude from a single model that a data point is in fact redundant. Therefore, several models with varying complexity need to be constructed to determine whether the data point is consistently not needed as a support vector. For each model the Lagrange multiplier values are inspected and if both Lagrange multiplier values of a data point are zero the data point is identified as a possible redundant point.

Next we determine the frequency at which a data point is suspected by the various models to be redundant. The higher the frequency, the more likely it is that the data point in question is in fact redundant. However, in our research we observed in the plot of the frequencies a characteristic curve which levels off for increasing detection rates. An example of this phenomenon can be seen in Figure 6.7(b) where we used the example in Chapter 2 and constructed various SVMs using an RBF kernel with $\sigma = 0.2$ and $C = 200$. Unlike in the outlier detection approach, the redundancy

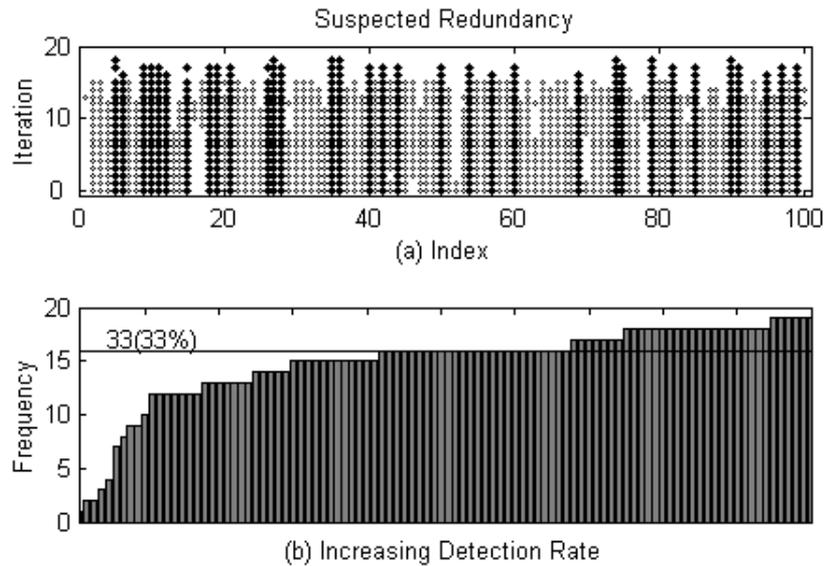


Figure 6.7: The frequency of suspected redundancy.

detection does not identify a single data point per iteration to be removed. That would be a very time consuming procedure. A larger number of data points can be identified at once by using the point where the redundancy frequency levels off or almost remain constant. All data points with frequencies equal or larger than the determined level are identified as redundant data points. In our example the level was determined to be 16 as indicated in graph (b). Also shown is the number of data points and percentage of data points this level identifies as redundancies. In Figure 6.7(a), the suspected redundancies per iteration are shown and the identified redundancies are indicated as well. In Figure 6.8 the predictions of the different SVM

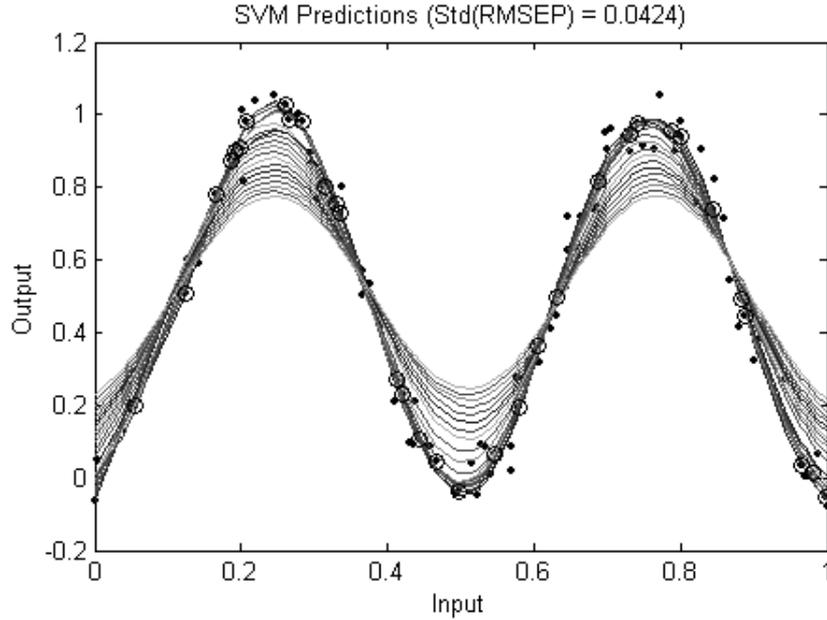


Figure 6.8: Predictions of various SVM models and detected redundant data points.

models are shown and the detected redundant data points are encircled as well.

One precautionary remark on applying the technique recursively. Removing *all* non-support vectors could seriously affect the data density of the input space and consequently the smoothness of the subsequent SVM. Furthermore, in the presence of large noise levels and some unknown outliers, removing all non-support vector data points would leave a data set which is largely determined by the noise and outliers. In turn, rebuilding a SVM based on such a data set could lead to a worse model.

Algorithm I. Redundancy Detection

- Step 1:** Steps 1-4 of Algorithm H.
- Step 2:** Set redundancy index matrix Red_I as an $\ell \times it_num$ matrix of zeros.
- Step 3:** For $k = 1$ to it_num
 run Algorithm B using $\epsilon(k)$
 identify possible redundancy indexes as:
 $I = \{i | \mathbf{w}_i = 0, i = 1, \dots, \ell\}$
 set $Red_I(I, k) = 1$
 end
- Step 4:** Calculate the frequency of indexes in Red_I :

```

for  $i = 1$  to  $\ell$ 
     $Freq(i) = \sum_{j=1}^k Red_I(i, j)$ 
end

Step 5: Calculate the largest frequency:  $M\_Freq = \max_i Freq$ .

Step 6: If  $M\_Freq > 10$ ,
    determine point where the frequency levels off:
        sort  $Freq(i)$  in increasing order
        calculate frequency  $f$  with longest sequence of constant frequency
        set cut-off level:  $cut\_off = f$ 
    find indexes of redundancies:
         $index = r = \{i | Freq(i) \geq cut\_off, i = 1, \dots, \ell\}$ 
    remove  $\mathbf{x}(i)$  and  $y(i)$  from data set
    repeat Steps 2-10
else
    stop
end

```

6.4 Industrial example

We show the efficiency of the SVM based redundancy detection on the same industrial example used in Section 5.4 of Chapter 5. In Figure 6.9 the frequency information is given. Note the characteristic levelling-off of the frequency for increasing detection rates.

In the first iteration a total of 84 data points were removed. A second iteration removed 108 data points. The third and final iteration removed 192 data points. The reduced data set after each iteration can be seen in Figure 6.10. The predictive capability of a SVM using the same parameters but based on a reduced data set is given as well. Note the the data set can be compressed to less than 50% of its original size without losing any predictive capabilities.

6.5 Conclusion

Let us consider the design requirement again:

Applications such as redundancy detection and outlier detection are important requirements of the development of inferential sensors.

The traditional approaches for outlier and redundancy detection are distance or density based. Recently it has been argued that such approaches are often unsuitable for high-dimensional data. Therefore in recent years scientists expressed the need for model-based approaches.

It has been observed by many that the SVM method could be used for detecting outliers as well as data compression. This is because the SVM method identifies

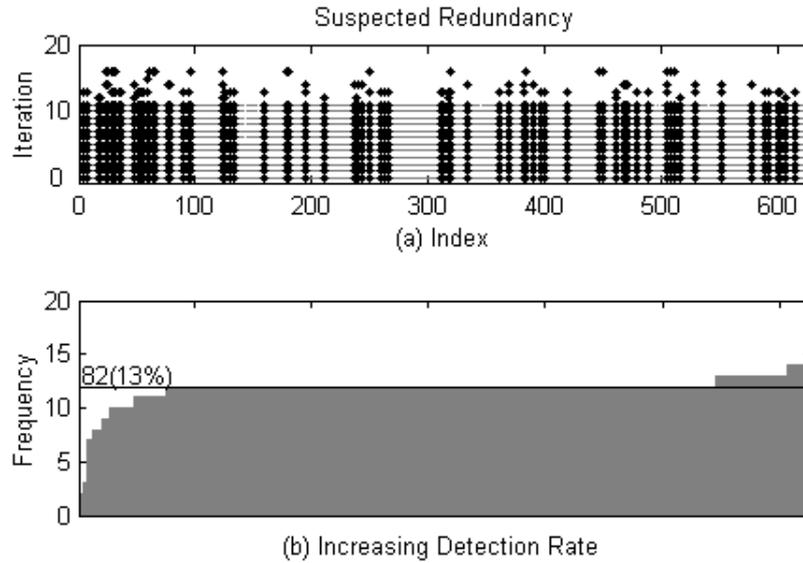


Figure 6.9: Frequency rates for industrial data set.

specific learning data points of interest that are then used to define the model. The SVM method has therefore the unique property that the magnitude of the weights can be used to identify outliers and redundancy. In this chapter we investigated the possibility of using the characteristics of SVMs as a basis for model-based outlier detection and redundancy detection approaches.

In the approach for outlier detection, we consider a data point to be a possible outlier when its corresponding Lagrange multiplier values hit the upper boundary of the constraint and it has a large error. A data point is identified as an outlier if it is repeatedly selected as a possible outlier by various SVM models. After removing this outlier, the procedure is repeated until no more outliers are identified. The algorithmic pseudocode for the outlier detection based on SVMs is given in Algorithm H. We also illustrated in an example the effectiveness of the proposed approach.

The redundancy detection approach inverts the reasoning of the outlier detection approach. Only now data points with weights that are repeatedly zero are suspected to be redundant. A data point will be considered as redundant when its frequency lies above the level where the frequency rate levels off. Therefore data points that are almost never used as support vectors by various SVM models are considered to be redundant. In Algorithm I the algorithmic pseudocode for the redundancy detection based on SVMs is given and applied to an industrial data set which was compressed to less than 50% of its original size without loss of quality in the model.

Finally, using the approaches presented in this chapter the inferential sensor is able to comply with the design specification as stated.

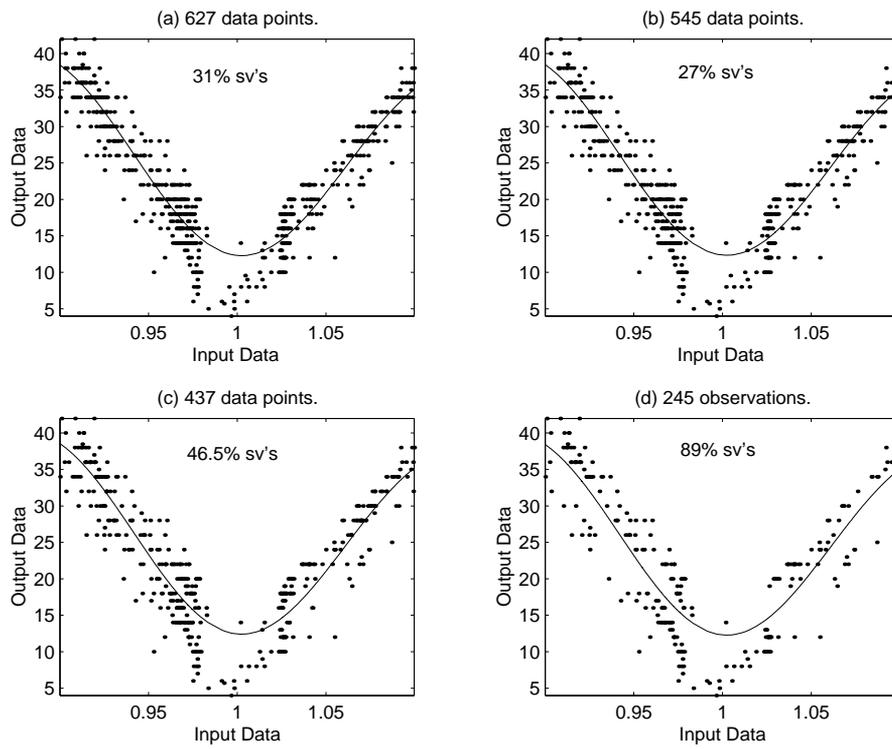


Figure 6.10: Compression of an industrial data set in three iterations.

Chapter 7

Incorporate Prior Knowledge

7.1 Introduction

As processes get more complicated and more research is being done on these processes, more information about the physical laws, constraints and conditions are becoming available. Any information about the learning task that is available in addition to the learning data is considered to be prior knowledge. Generally speaking, it is only possible for data-driven models to generalise from the training data to unseen test data if prior knowledge was included during the learning process [65]. Therefore, the new generation of inferential sensors should not only be built on empirical data but also incorporate any other available information.

The incorporation of prior knowledge has been the object of research in many fields. The easiest way to incorporate human knowledge is through fuzzy logic [113],[114] and rule-based systems [14],[95]. The systems strongly depend on a priori knowledge and are aimed at providing a model with modes of human reasoning that are approximate rather than exact [39],[95]. However, successful fuzzy applications are limited to rather low-dimensional models [11].

Unfortunately, incorporating prior knowledge or information in data-driven models is not easy [91]. There are various kinds of prior knowledge to be considered [78]. One well-known form is *feature selection*. Here knowledge about the correlation between dimensions is used to extract those features that cause the observed variation [11]. Knowledge about the probabilistic models generating the data can be incorporated into the kernel as is done in Fischer kernels [30]. Another form is by making use of smoothness assumptions of the problem [92]. For example a Bayesian maximum-a-posteriori setting which corresponds to a smoothness prior of $\exp(-\|\gamma f\|^2)$. Prior knowledge could also be considered as knowledge of certain transformations of the input data known to leave the function values unchanged. This type of prior knowledge is known as invariance information and mainly applicable for image processing [77],[8]. A full discussion can also be found in [78]. Often the user wants an understandable model. Learning machines like NNs and SVMs lack the ability to present the model in a closed-form formula. However, this problem can partly be solved through semi-parametric modelling where a linear combination of specific parametric components

is added to the objective function. This requires prior knowledge about the likelihood of specific parametric components to be part of the solution [78].

All the forms of prior knowledge mentioned above require either some knowledge of the probabilistic nature of the problem or the interdependency of the variables. Often in chemical plants neither are available. What is available is the boundary conditions for processes, information about the varying levels of noise in the input space and the sensitivity of the variables.

In this chapter three forms of prior knowledge frequently encountered in the chemical industry are investigated. In the first section the incorporation of boundary information is considered. This type of information is only useful if the resulting model is able to extrapolate well. The second section investigates the incorporation of sensitivity analysis information obtained from the dimensionality reduction step for multi-dimensional scaling purposes. This approach takes feature selection one step further. After the relevant features are selected, the sensitivity of the relevant features are taken into account. Finally in the third section, a procedure for dealing with heteroscedastic noise is discussed. Normally, it is assumed that in Support Vector Regression the size of the tube remains constant throughout the input space and therefore, it is also assumed that the noise level remains constant. By varying the size of ϵ the tube can take an arbitrary shape.

7.2 Boundary information

From physical laws and constraints information about the behaviour of a process at the boundaries can be derived. The idea is to make use of this information during the learning process in order to build more intelligence into the resulting inferential sensor. The challenge is how to present the information to the learning machine. Since many of the learning methods learn from a set of data points, the obvious solution is to create observations that reflect the boundary conditions.

These artificially created observations are expected to be outside the input space of the actual observations. Therefore, it is to be expected that the learning machine will have to extrapolate between the true observations and the supplied boundary information. To make use of the extra information available, the resulting model must have a good generalisation ability. This is where the mixed kernel approach for support vector machines becomes very useful, for it is able to fully exploit the prior information added to the set of observations.

Adding prior information into the kernel in the form of boundary conditions of the input data increases the performance of the mixed kernels even more. To demonstrate this, we added boundary conditions by supplying the SVM with only two points, each on the edge of the input space.

In Figures 7.1 and 7.2 we show the performance of polynomial kernel SVMs and RBF kernel SVMs using prior knowledge. We see that in both cases the SVMs are much better. However, there is still no single kernel parameter for either polynomial or RBF kernels that will result in a model with both good interpolation and extrapolation abilities.

In Figure 7.3 the performance of the mixed kernel is given. Although it is not right

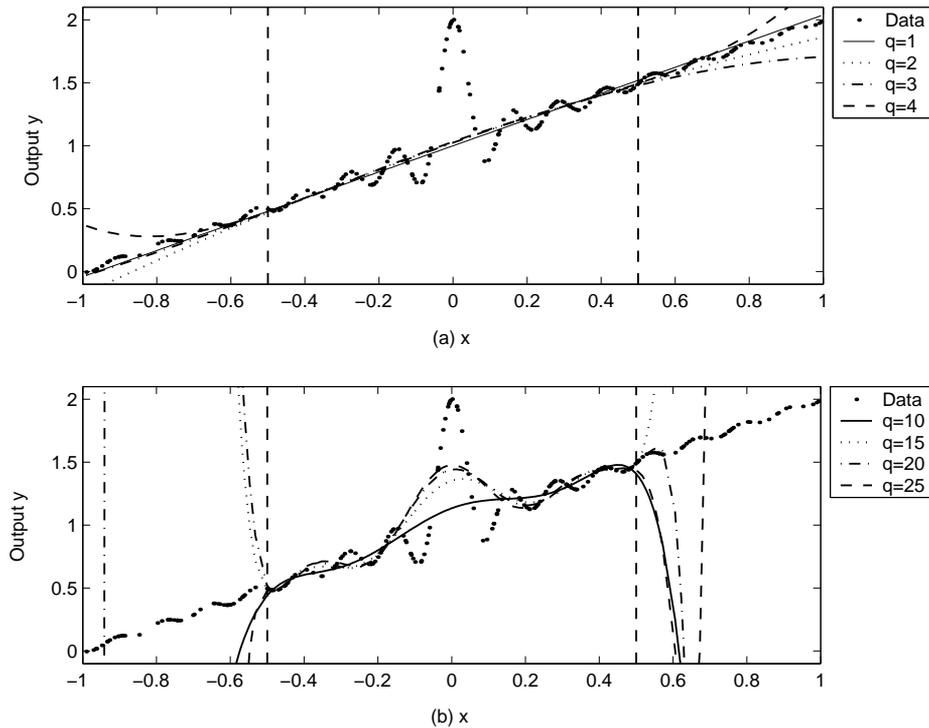


Figure 7.1: Polynomial kernels with Prior Information (a) for degrees $q = \{1, 2, 3, 4\}$ (b) for degrees $q = \{10, 15, 20, 25\}$.

on the mark, it is still much better in terms of the balance between interpolation and extrapolation performance.

We see that using prior knowledge increases the generalisation ability of all SVM models. Other forms of prior knowledge incorporation are discussed in [77].

7.3 Multi-dimensional scaling

The kernel value of a typical kernel function is the inner product of two n -dimensional vectors \mathbf{x} and \mathbf{z} and is determined as

$$K(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^n \theta(x_i) \cdot \theta(z_i). \quad (7.1)$$

In the kernel function in (7.1) each dimension has equal importance and contribute equally to the kernel value. In real life problems, this assumption may be invalid.

One way to address this is to assign a weight to each dimension. Thus giving the dimensions a relative importance with respect to each other. Fortunately, the

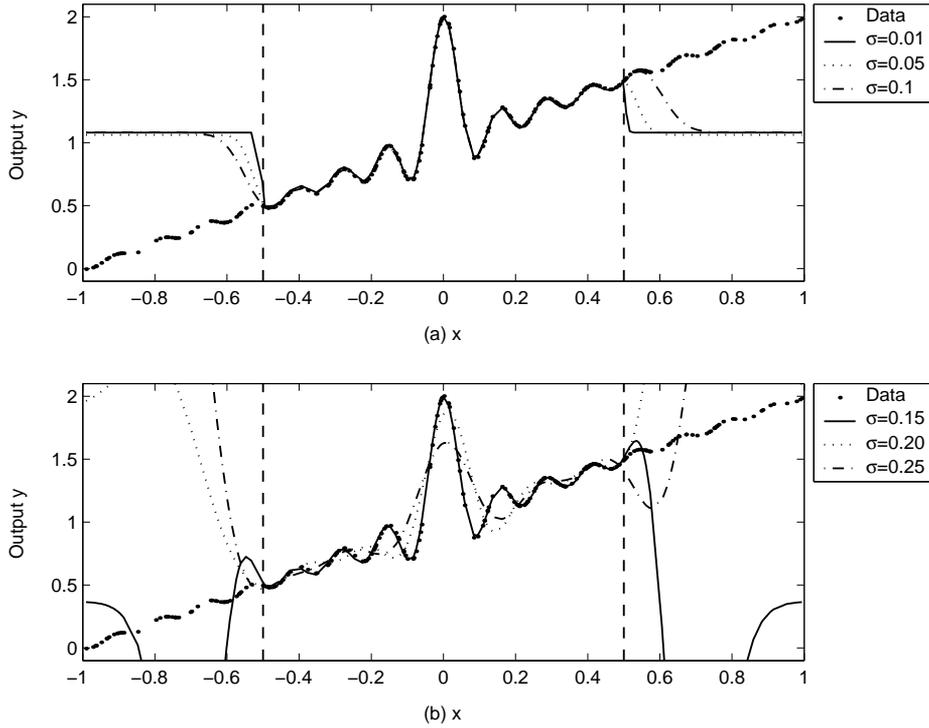


Figure 7.2: RBF kernels with prior information (a) for widths of $\sigma = \{0.01, 0.05, 0.1\}$ (b) for widths of $\sigma = \{0.15, 0.2, 0.25\}$.

formulation of the kernel function can easily be altered such that a weight is assigned to each dimension. For a given set of weights

$$\mathbf{w} = (\omega_1, \omega_2, \dots, \omega_n), \quad (7.2)$$

a weighted kernel function can then be written as

$$K(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^n \omega_i (\theta(x_i) \cdot \theta(z_i)), \quad (7.3)$$

Each weight thus scales the corresponding dimension's contribution to the kernel function. Such an approach we define as multi-dimensional scaling ¹.

Information obtained from sensitivity analysis of the input dimensions can be used in the support vector machine to perform multi-dimensional scaling. Sensitivity

¹There is a difference between our definition multi-dimensional scaling and the multivariate statistical method MDS. In multivariate statistics MDS is a technique that is designed to construct a 'map' which shows the relationships between a number of objects, given only a table of distances between them [52].

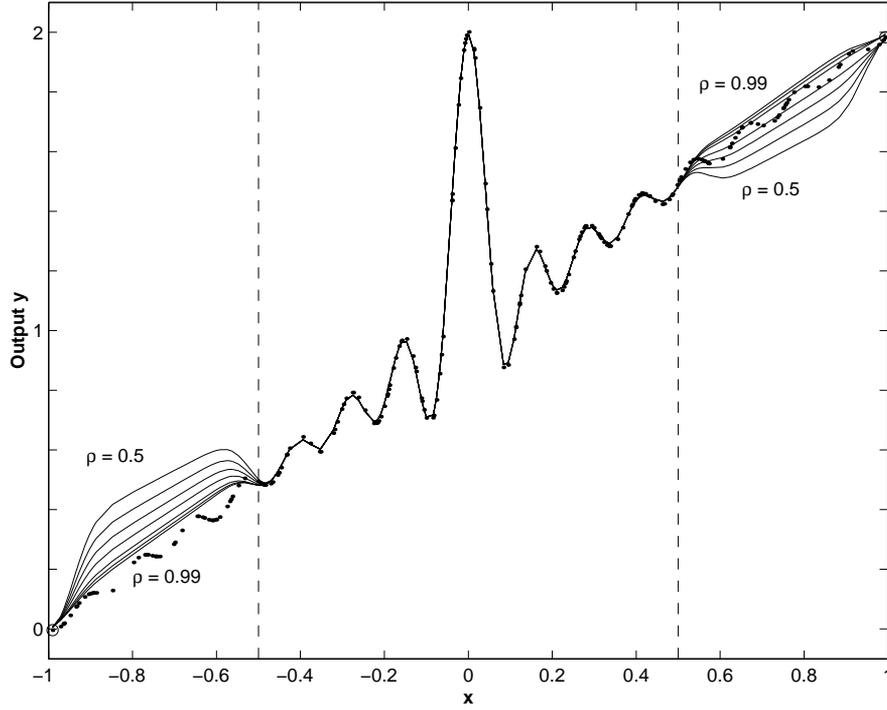


Figure 7.3: Mixed kernels with prior information of first degree polynomial, $\sigma = 0.05$ width of RBF and $\rho = \{0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99\}$.

analysis of the input dimensions is usually used to determine which dimensions are relevant to the problem [14]. For multi-dimensional scaling purposes, we use the sensitivity of the input dimensions as an indication of the relative importance of each dimension. In industry stacked NNs or GP are used for sensitivity analysis [41]. In practice the stacked NN approach is faster and the results are better reproducible [85].

The values of the weights are often percentages or ratio's. In order to retain the original kernel formulation for instances with equal weights, the weights should first be normalised such that $\sum_{i=1}^n \omega_i = n$. The normalisation is done using

$$\omega_i = \frac{n * \omega_i}{\sum_{i=1}^n \omega_i}. \quad (7.4)$$

To show the advantage of using multi-dimensional scaling, we use a six-dimensional industrial data set. In Figure 7.4 the output data of the learning data, indicated by a dot, as well as the test data, indicated by a plus, is shown for each dimension. The input data were range scaled to $[0, 1]$. Note that the test data differ in several dimensions from the learning data. Therefore, any model resulting from the learning data will have to extrapolate when predicting the test data.

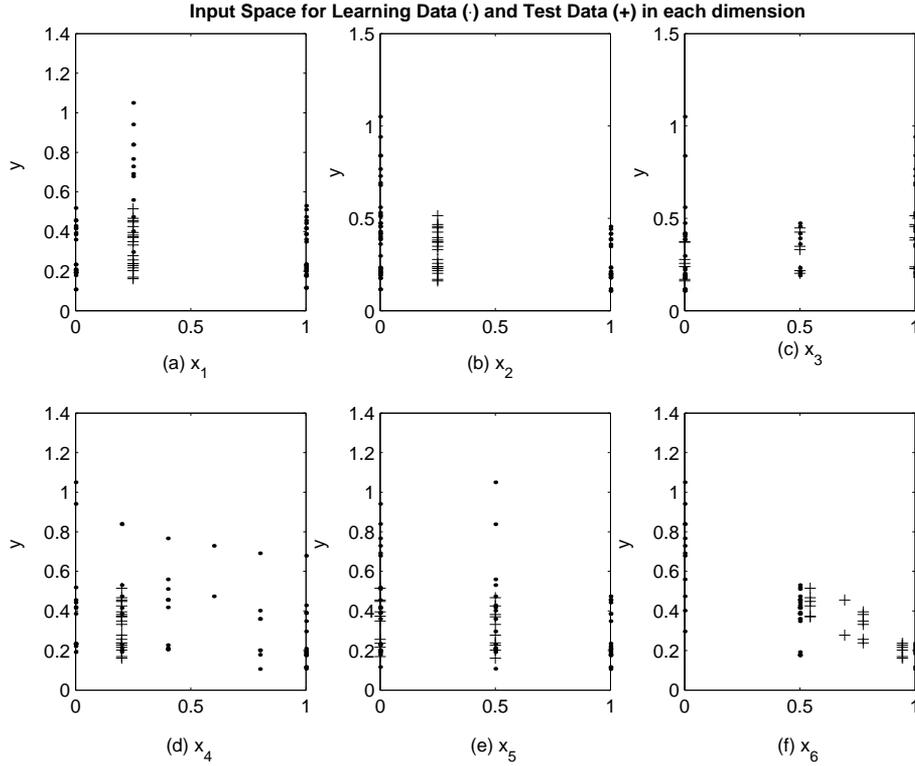


Figure 7.4: Input space of data with respect to each dimension.

The weights $\omega = (0.3, 0.44, 0.72, 1, 0.5, 1)$ were obtained from a NN sensitivity analysis of the input dimensions. Three types of kernels, namely RBFs, polynomials and a mixture of RBFs and polynomials, were used. The optimal settings for the kernel parameters, ϵ and C were then obtained using Algorithm D and the L-curve method, respectively. In Table 7.1 the parameter settings used in the different SVM models are given, as well as the percentage support vectors. The RBF kernel needed a σ -value of 1.0 to be able to predict the test data. Since the input data were range scaled to $[0, 1]$, a RBF kernel with $\sigma = 1.0$ represents a global kernel that takes into account all data points in the learning range. That confirms that the resulting model will indeed have to extrapolate in order to predict the test data. In each case where the multi-dimensional scaling was used, the number of support vectors decreased.

Next we investigate the effect on the performance of the SVM models with and without the use of multi-dimensional scaling for the different kernel functions. In Figure 7.5 four statistical measures are used to show the effect of multi-dimensional scaling. These measures are the correlation coefficient, standard deviation, relative error and R^2 -statistic. The statistical performance of models built using three types of kernels (RBF, polynomial and a mixture of the two) are compared. The graphs

Table 7.1: Parameter settings and the percentage support vectors.

Kernel Function	Kernel Parameters	ϵ	C	Sensitivity Used	% SV's
RBF	$\sigma = 1.0$	0.05	156	No	51.6
RBF	$\sigma = 1.0$	0.05	156	Yes	40.3
polynomial	$d = 2$	0.05	2	No	43.5
polynomial	$d = 2$	0.05	3	Yes	40.3
mixture (poly,RBF)	$\rho = 0.88$ $d = 2, \sigma = 0.4,$	0.05	2	No	33.9
mixture (poly,RBF)	$\rho = 0.88$ $d = 2, \sigma = 0.4,$	0.05	2	Yes	30.6

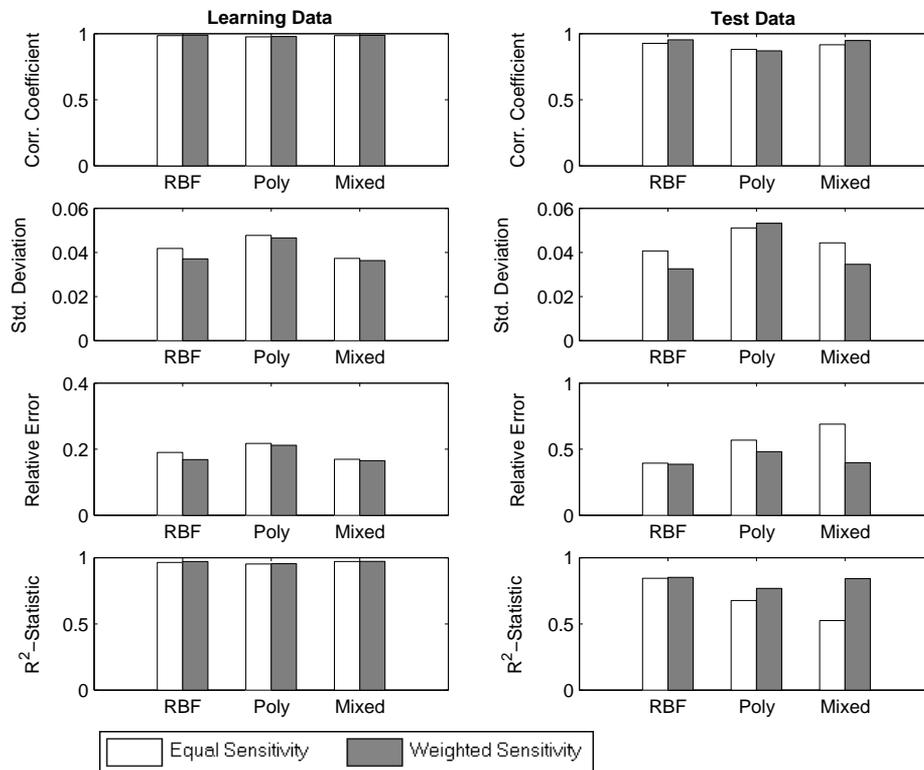


Figure 7.5: Effect of multi-dimensional scaling on the error statistics of SVMs of different kernel functions.

also show the difference in performance between the learning and test data. The error statistics for the learning data are shown on the graphs to the left and those for

the test data are shown in the graphs on the right. The white bars show the error statistics of the models using no multi-dimensional scaling, whilst the grey bars show the error statistics of the models using multi-dimensional scaling. From the figure one can see that the performance of the model in predicting the learning data slightly improves when multi-dimensional scaling is used. The positive effect of using multi-dimensional scaling is more evident in the performance of the model using the test data.

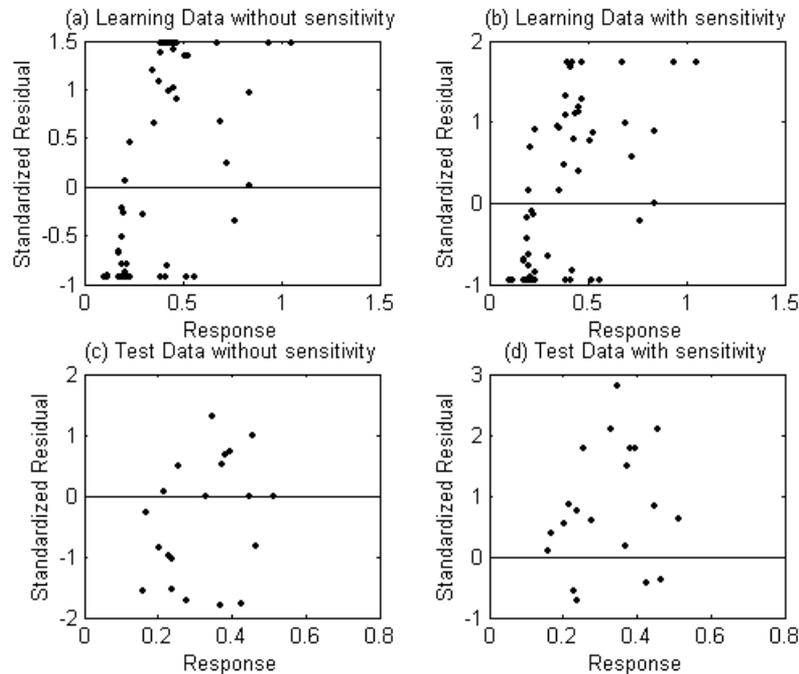


Figure 7.6: Effect of multi-dimensional scaling on the standardized residuals of SVMs using the RBF kernel.

Finally, we show the effect that multi-dimensional scaling has on the standardized residuals for the three types of kernels in Figures 7.6, 7.7 and 7.8. In all three figures the graph (a) shows the standardized residuals of the prediction of the learning set using the model without multi-dimensional scaling. The Graph (b) shows the standardized residuals of the prediction of the learning data using the model with multi-dimensional scaling. Graph (c) depicts the residual of the test data predicted by the model without multi-dimensional scaling. The last graph, Graph (d), in each figure plots the residual of the test data predicted by the model using multi-dimensional scaling. Note that in all three cases the residuals for both the predictions of the learning and test data becomes more balanced and evenly spread when multi-dimensional scaling is used during the modelling. Based on the observations supplied by the error statistics,

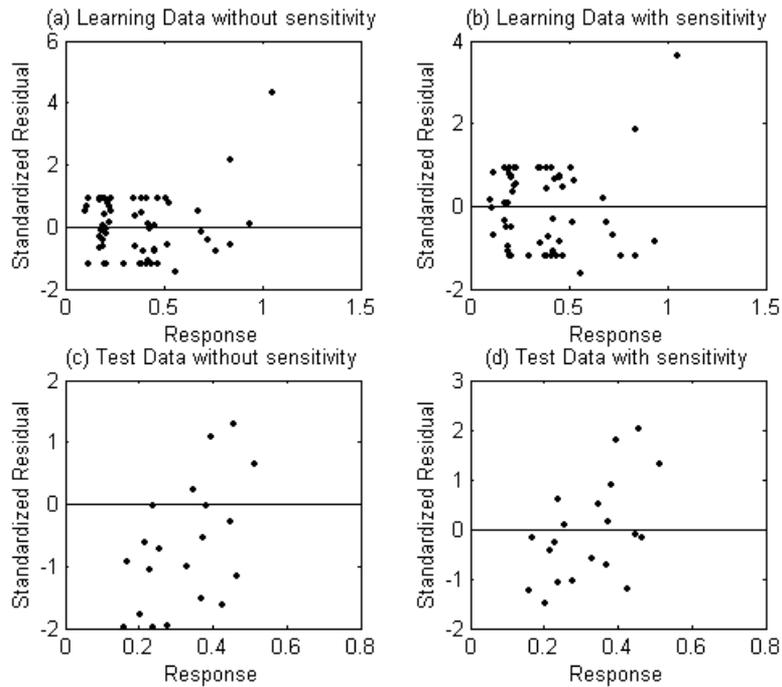


Figure 7.7: Effect of multi-dimensional scaling on the standardized residuals of SVMs using the polynomial kernel.

one can see a tendency that both the learning ability of the learning data and the predictive capability of the unseen test data is improved.

7.4 Variable ϵ

In classical SVM for regression it is assumed that the size of the tube remains constant throughout the input space. That means that it is implicitly assumed that the noise level remains constant throughout the input space. In industrial data sets this is not necessarily the case. Often some information about the dependency of different noise levels in the input space is known.

The SVM for regression can easily deal with this type of information by varying the size of ϵ such that the width of the tube can take a shape corresponding to the different noise levels [74],[78]. In the standard ϵ -SVM for a given vector $\epsilon = \epsilon_1, \dots, \epsilon_\ell$, the QP problem is as follows.

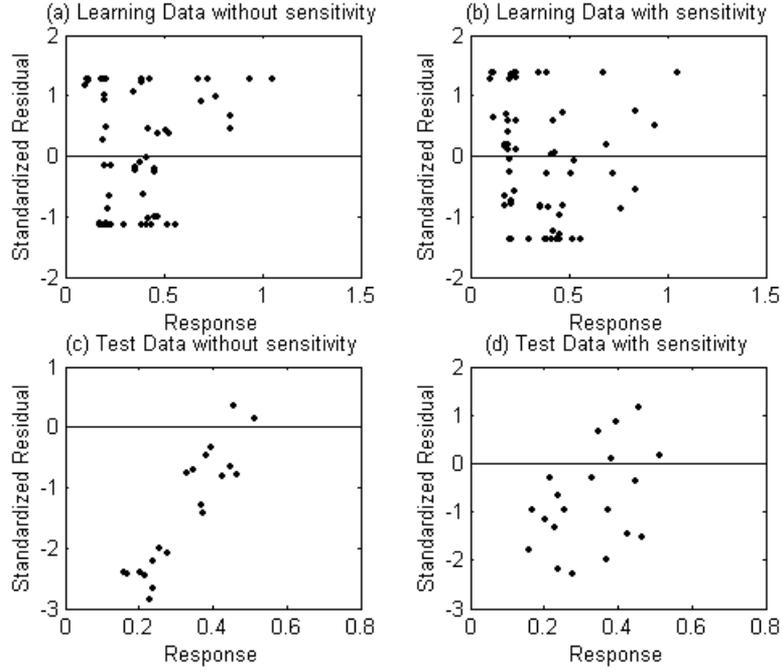


Figure 7.8: Effect of multi-dimensional scaling on the standardized residuals of SVMs using the mixed kernel.

$$\text{minimise } \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{\ell} \sum_{i=1}^{\ell} (\xi_i + \xi_i^*), \quad (7.5a)$$

$$\text{subject to } ((\mathbf{w}, \mathbf{x}_i) + b) - y_i \leq \epsilon_i + \xi_i, \quad i = 1, \dots, \ell, \quad (7.5b)$$

$$y_i - ((\mathbf{w}, \mathbf{x}_i) + b) \leq \epsilon_i + \xi_i^*, \quad i = 1, \dots, \ell, \quad (7.5c)$$

$$\xi_i, \xi_i^* \geq 0, \quad i = 1, \dots, \ell. \quad (7.5d)$$

Again through introducing Lagrange multipliers, finding the saddle points of the

Lagrangian and writing the Wolf Dual problem, we get

$$\begin{aligned} \text{maximise} \quad & \sum_{i=1}^{\ell} y_i (\alpha_i^* - \alpha_i) - \sum_{i=1}^{\ell} \epsilon_i (\alpha_i^* + \alpha_i) \\ & - \frac{1}{2} \sum_{i,j=1}^{\ell} (\alpha_i^* - \alpha_i) (\alpha_j^* - \alpha_j) K(\mathbf{x}_i, \mathbf{x}_j), \end{aligned} \quad (7.6a)$$

$$\text{subject to} \quad \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) = 0, \quad (7.6b)$$

$$0 \leq \alpha_i, \alpha_i^* \leq \frac{C}{\ell}, \quad i = 1, \dots, \ell. \quad (7.6c)$$

The best approximating function or model still has the form

$$f(\hat{\mathbf{x}}) = \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) K(\mathbf{x}_i, \hat{\mathbf{x}}) + b, \quad (7.7)$$

where b is the determined bias.

To show the advantage of using a variable ϵ value, we use the industrial example in Section 5.4 of Chapter 5. In Figure 7.9(a), predictions of the data are shown using a SVM with constant ϵ set at 5, $C = 15000$ and a mixture of a RBF of width 0.15 and polynomial kernel of degree 1, and a mixing coefficient of 0.98. In graph (a) of Figure 7.9 there is noticeably less data in the region between 0.4 and 0.6. It is clear that majority of the data points within this region lies below the model. However, by decreasing the value of ϵ in this region the model can be forced to go through the lower points. In a simple implementation we use ϵ values of 2 in the region between 0.4 and 0.6 and elsewhere 5. Figure 7.9(b) shows the predictions of the data using different values for ϵ . The model now clearly fits the data within this region better.

In practice it might be a problem to determine the vector of ϵ values. A more appropriate approach may be to use a function of the noise instead in addition to the ν -SVM [78].

Let $\zeta_k^{(*)}$ with $k = 1, \dots, p$ be a set of $2p$ positive functions on the input space X . For given $\nu_i^*, \dots, \nu_p^* \geq 0$, the QP problem is

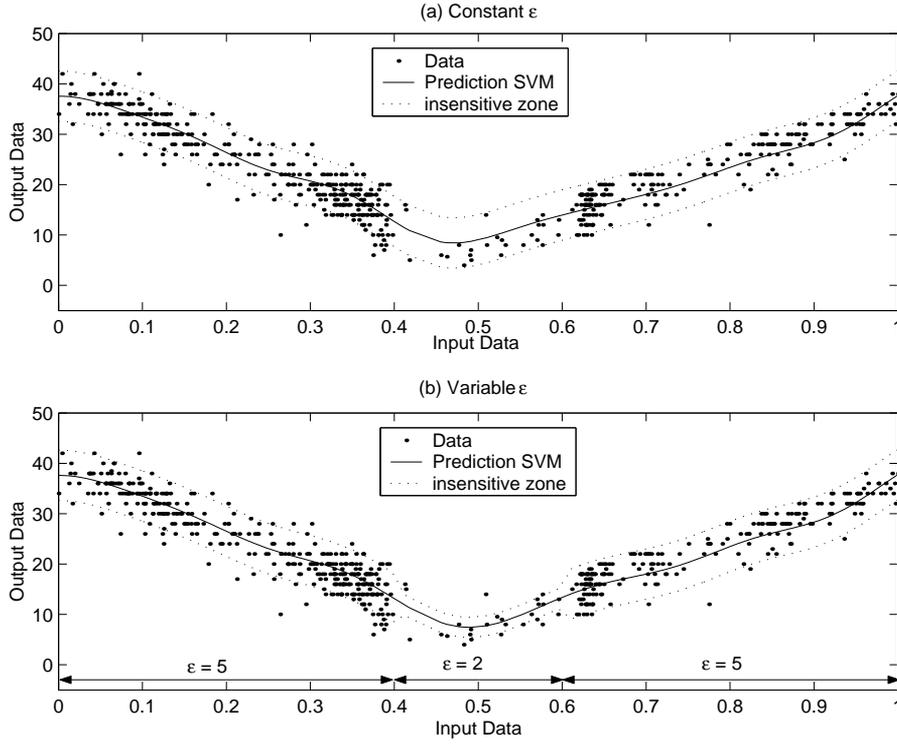
$$\text{minimise} \quad \frac{1}{2} \|\mathbf{w}\|^2 + C \cdot \left(\sum_{k=1}^p (\nu_k \epsilon_k + \nu_k^* \epsilon_k^*) + \frac{1}{\ell} \sum_{i=1}^{\ell} (\xi_i + \xi_i^*) \right), \quad (7.8a)$$

$$\text{subject to} \quad ((\mathbf{w}, \Phi(\mathbf{x}_i)) + b) - y_i \leq \sum_{k=1}^p \epsilon_k \zeta_k(\mathbf{x}_i) + \xi_i, \quad i = 1, \dots, \ell, \quad (7.8b)$$

$$y_i - ((\mathbf{w}, \Phi(\mathbf{x}_i)) + b) \leq \sum_{k=1}^p \epsilon_k^* \zeta_k^*(\mathbf{x}_i) + \xi_i^*, \quad i = 1, \dots, \ell, \quad (7.8c)$$

$$\xi_i^{(*)} \geq 0, \quad i = 1, \dots, \ell, \quad (7.8d)$$

$$\epsilon_k^{(*)} \geq 0, \quad k = 1, \dots, p. \quad (7.8e)$$

Figure 7.9: Example of the advantage of using variable ϵ values.

Again through introducing Lagrange multipliers, finding the saddle points of the Lagrangian and writing the Wolf Dual problem, we get

$$\text{maximise} \quad -\frac{1}{2} \sum_{i,j=1}^{\ell} (\alpha_i^* - \alpha_i) (\alpha_j^* - \alpha_j) K(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) y_i, \quad (7.9a)$$

$$\text{subject to} \quad \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) = 0, \quad (7.9b)$$

$$0 \leq \alpha_i^{(*)} \leq \frac{C}{\ell}, \quad i = 1, \dots, \ell, \quad (7.9c)$$

$$\sum_{i=1}^{\ell} \alpha_i^{(*)} \zeta_k^{(*)}(\mathbf{x}) i \leq C \cdot \nu_k^{(*)}, \quad k = 1, \dots, p. \quad (7.9d)$$

Note that the optimisation problem is still linear in the α 's. The problem can be reduced to the original ν -SVMs by setting $p = 1$, $\zeta^{(*)} \equiv 1$, and omitting the $\nu_k^* \epsilon_k^*$ term in the primal formulation.

By using different values of ν and ν^* , the tube is not symmetric. However, the advantage of using the same ν on both sides of the tube is that ϵ and the bias, b , can be determined in the same way as in the original formulation.

7.5 Conclusion

Let us consider the original design requirement again:

It is required that the new generation of inferential sensors should try to incorporate any other form of prior knowledge that is available.

The incorporation of prior knowledge is one of the most challenging task in inferential sensor learning. Prior knowledge can take many forms and can be entered in a number of different ways. In this chapter three possibilities of incorporating prior knowledge were discussed.

One form of prior knowledge is boundary information or the behaviour of a process in the limit. This type of prior knowledge can only be useful if the model is able to extrapolate effectively. Through the introduction of the mixed kernel approach in SVMs, the resulting model has gained the ability to extrapolate conservatively outside the known learning space. It was shown in a number of examples that the inferential sensor built on support vector machines using boundary information can predict well outside the known learning space. Furthermore, being able to use low degrees of polynomials, the model can gracefully degrade which is far better than simply leveling off to a bias (like for a RBF kernel alone) or becoming virtually unbounded (like for high degrees of polynomials).

The other form of prior knowledge introduced is the multi-dimensional scaling approach where the relative importance of relevant features is incorporated. The relative importance is determined by other types of learning methods, like NNs or Genetic Programming. This determination of relative importance is often also called sensitivity analysis and its explanatory power is an important asset. The information is incorporated through the kernel functions. It was shown that although the performance of the model with respect to the learning capacity is only slightly improved, the ability of the model to extrapolate using the test data is increased dramatically.

The last form of prior knowledge considered is the variable size of the ϵ -insensitive zone. The type of prior knowledge is focussed on *a priori* information on the learning space. In case where the density of observation is low or sparse, the value of ϵ in that region can be set small so that the model is forced through these rare observations and incorporated them as support vectors. Determining the different values of ϵ might prove to be difficult. Another way to achieve the variable tube size, is to use the ν -Support Vector Machine with supplied functions $\zeta(\mathbf{x})$ which depends on the input space. The functions could reflect the knowledge of different noise levels on the input data.

Of course there are many other forms of prior information. For example, first principle models. How to incorporate the information intelligently and with ease, remains virtually an untouched research field. What remains is that the inferential sensors, built on incorporated prior knowledge, all have increased learning and generalisation abilities.

Part III

Operational Requirements

Chapter 8

Adaptivity

8.1 Introduction

Many modelling approaches deal with dynamic processes. For example, due to the slow degradation of a catalyst bed, the operating conditions change over time. After some time, the reactor has to be shut down to replace the catalyst. The shut down of a reactor is scheduled at fixed time intervals. However, since the catalyst bed does not degrade every time at the same rate, the shut down may often happen too soon or too late. If the shut down is too soon, the unused catalyst is wasted. If the shut down happens too late, too much off-spec product is manufactured.

In online applications an inferential sensor frequently becomes invalid after some time since the model is incapable of predicting well under such changing conditions. Therefore, it is necessary for the inferential sensor to adapt to new information or conditions [112]. A number of levels of adaptation is proposed and discussed in this chapter of the design thesis.

To achieve adaptivity, the inferential sensor first needs to know when and if something novel has occurred. It is not a question of recognising obvious changes in the process like new equipment or procedures. The problem is about detecting subtle changes like seasonal behaviour or the slow degradation of a catalyst. This chapter also discusses and illustrates a new approach for detecting novelty. This new approach involves the characteristics of the support vectors determined by a SVM.

The ultimate goal is that the adaptation of a inferential sensor occurs online [112],[70]. One way to achieve this is to use transductive inference [19],[106]. That is to build each time a model for a specific point of interest and predict the outcome of this point. Thereafter, the model is discarded. This idea was also explored in [61] and [62] where an adaptive online learning algorithm is proposed.

The layout of this chapter is as follows. In the first section we illustrate a support vector based approach for novelty detection. A number of adaptation levels are discussed in the second section. Finally, in the last section of this chapter the issues and implementation of the transductive modelling approach are discussed.

8.2 Novelty detection

Novelty in terms of chemical applications are those observations related to changed conditions in the plant or processes. The ability to detect novelty or being aware of possible novel information is very important in inferential sensor design. Currently, novelty is detected based on the error statistics of the inferential sensor [112].

The support vector machine method provides a different approach in detecting novelty. A support vector approach for novelty detection has been applied to classification problems [80],[9],[25],[78]. Again very little has been done for regression applications [33]. In our approach we use the fact that support vectors are those observations that are difficult to predict. When a new observation is selected as a support vector, there is a high probability that something new is happening.

Of course one cannot make the decision based on just one observation and one support vector machine model. However, if a new observation is repeatedly chosen as a support vector by several models over a period of time, there is a high probability that something novel has occurred.

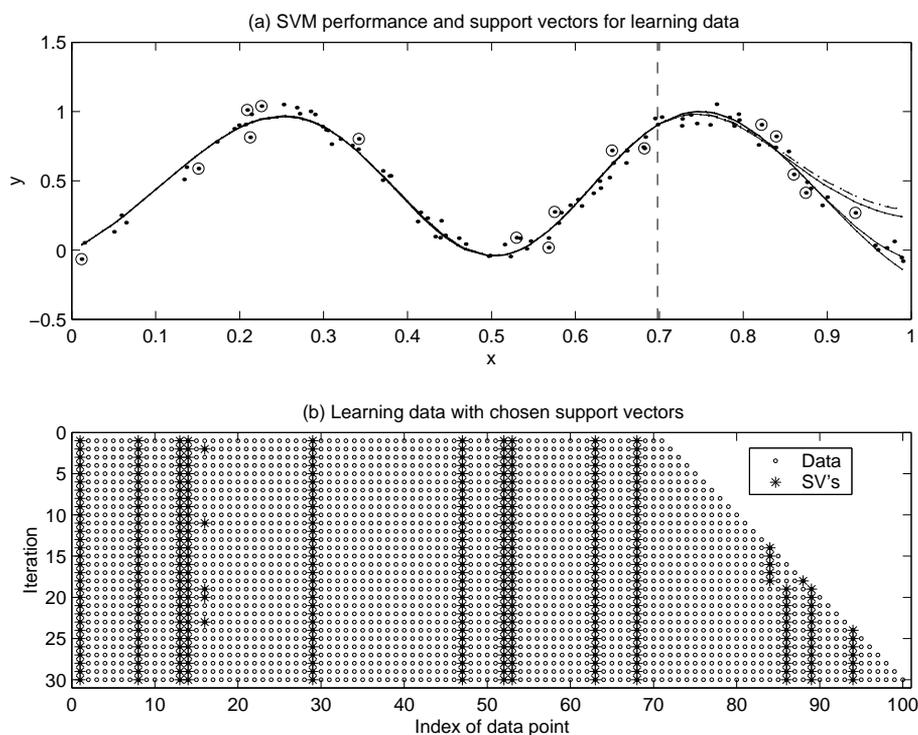


Figure 8.1: Novelty detection using SVMs.

The process of novelty detection using support vector machines is illustrated in Figure 8.1. In graph (a) the performance of the SVM models is given for the learning data as well as the support vectors used. The vertical dashed line indicates the starting

point of the novelty detection routine. All learning data before this line, is known at the first iteration. In the following iterations each time one data point is added to the learning set. Graph (b) shows at each iteration which data points were used as learning data, indicated by crosses, and which of those data points were subsequently selected as support vectors, indicated by dots. From Figure 8.1 it is clear that the previously chosen support vectors do not change over time. Furthermore, the newly selected support vectors also stay stable after a short period of time. Therefore, if a new observation is selected as a support vector there is a high probability that something is changing.

Finally, the ability of an inferential sensor to detect novelty fast and accurately is an important aspect needed to meet the design requirement of adaptivity required for inferential sensor development.

8.3 Adaptation levels

The learning data may not always include samples from all plant operating regions. Something in the plant's operating state may change to one that was not included in the original learning set. The inferential sensor needs to be able to adapt to that [112]. The adaptation of inferential sensors can take place in several ways, like switching to another known model when the process enters a new operating regime, changing the parameters of the current inferential sensor or even completely rebuilding the inferential sensor offline [33]. Adaptation does not only imply the change of the model used by the inferential sensor, but also the operational state of the inferential sensor. For example putting the inferential sensor on alert for detected novelty.

Since the inferential sensor can be adapted in different ways, one could grade the different forms of adaptation in levels of severity [33]. It could be that no novelty has been detected and the model is still operating satisfactory. No adaptation is necessary. The only thing needed to be done is to update the history sets with the current observation. This is a Level 0 adaptation.

At a next level the process could be drifting off into an unknown domain, but not enough information is known yet to make any adjustment. This occurs when the inferential sensor detects a novelty, but the effect of noise or the presence of an outlier could not be ruled out as the cause of the detected change. The adaptive inferential sensor will therefore not change until more is known. However, the inferential sensor will be alerted to the possible occurring novelty. In this adaptation level not only the history set is updated with the current observation but also the learning data set. The adaptation that occurs here is of Level 1.

The inferential sensor may also have detected that the current process is running in another known operating domain. Therefore, a switch to another known model is triggered. This type of adaptation is a Level 2 adaptation. A third level of adaptation occurs when not a single model is used but a combination of known models. This level of adaptation often occurs when a process is switching from one operating regime to another.

Higher levels of adaptation will require a partial or complete rebuilding of the inferential sensor. In a Level 4 adaptation, for example, the inferential sensor is built

online, like in the transductive modelling approach. For each observed data point a model is built with the sole purpose of predicting that observation. Thereafter, the history and learning data sets are updated with the observation. In this level, each data point is seen as a novelty. To rebuild a inferential sensor partially, is a Level 5 adaptation. The adaptation typically involves the adjustment of some parameters of the inferential sensor. Here the learning data set is largely still valid and only a small adjustments needs to be made. Finally, a Level 6 adaptation is the most severe form of adaptation which requires the inferential sensor to go off-line and restart the whole modelling process. This would occur when the process is operating in a completely unknown domain to such an extent that the learning data set is completely invalid. The results of the lower level adaptations are sent back to the operational inferential sensor. An overview of all the levels of adaptation is given below.

Level 1: No adaptation.

Action: Update history sets.

Level 2: Novelty detected.

Action: Not enough known yet, wait for more data.

Level 3: Model disagreement.

Action: Switch to other model (if available).

Level 4: Single model invalid.

Action: Use combination of models.

Level 5: Transductive Modelling.

Action: Determine neighbourhood, build model, update history and learning data sets.

Level 6: Confirmed detection of novelty.

Action: Partially rebuild inferential sensor.

Level 7: Learning data set invalid.

Action: Rebuild inferential sensor offline.

8.4 Transduction

Process engineers in industry are increasingly relying on inferential sensors to monitor and control processes [40],[100]. They cannot wait for modelling scientists to adapt or rebuild a inferential sensor. They require that the whole process of building, monitoring and adapting of the inferential sensor be done online as much as possible [112].

Transductive modelling [19] is ideal for online applications. The whole idea is that for a given test point, a neighbourhood of data points is selected from the learning set. Using the selected neighbourhood as a learning set, a model is built to predict the output of the test point. In transductive modelling, the model is not necessarily stored or used again. For every new test point, a new model may be built.

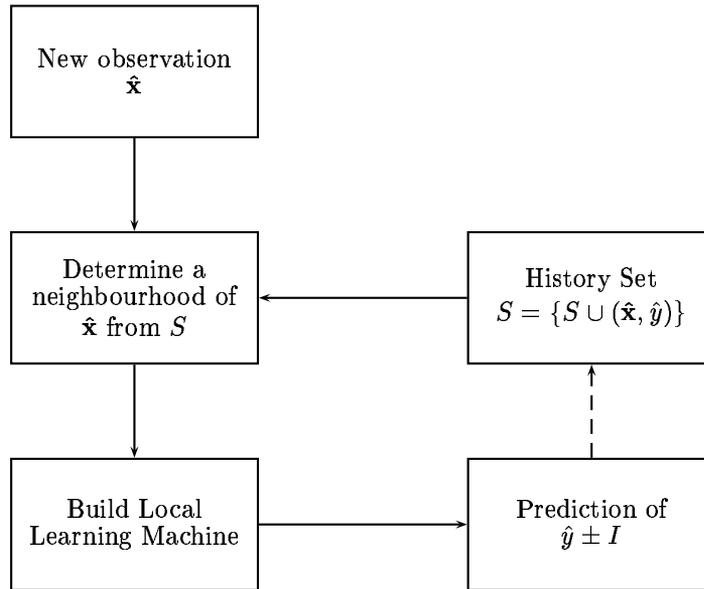


Figure 8.2: The Transductive Learning Machine

In Figure 8.2 a diagram of the transductive learning process is given. Assume that in any point in time an observation \mathbf{tx} is given. From the existing data set S of learning data, a neighbourhood around the observation \mathbf{tx} is determined. In order to select the neighbourhood a number of aspects has to be taken into account, namely the size of the neighbourhood, type of distance measure, data density, scaling, dimensionality, noise and outliers, and interpolation or extrapolation ability.

With respect to the size of the neighbourhood, one has to select enough relevant data points to build an appropriate model. This means that the density of data points in such a neighbourhood should be uniform in all directions, otherwise the neighbourhood will not be well balanced. If the neighbourhood is not well balanced, the resulting model may not be able to interpolate or extrapolate well. Therefore, the neighbourhood selection step is a very crucial part.

The way a neighbourhood is selected depends on the distance measure used. It is important to use an appropriate distance measure that takes into account the scaling of the data points and the dimensionality of the data set. Furthermore, the distance measure should also be robust with respect to noise and outliers present in the learning data.

The type of distance measure can vary from the straightforward Euclidian distance to multi-dimensional methods like clustering and Delaunay tessellations. In the next two sections, the Euclidian distance and the Delaunay tessellations method are discussed.

8.4.1 Euclidian distance

The Euclidian distance measure between two vectors is determined by [42]

$$d(\mathbf{x}, \mathbf{z}) = \sqrt{\sum_{i=1}^n (\mathbf{x}_i - \mathbf{z}_i)^2}. \quad (8.1)$$

The smaller the value of d , the closer the two data points are to each other. The neighbourhood of size k around the test point therefore consists of the k data points in the learning set that are the closest to the test point in terms of the Euclidian distance measure.

The size of the neighbourhood is an integer value k , which is set *a priori* by the user.

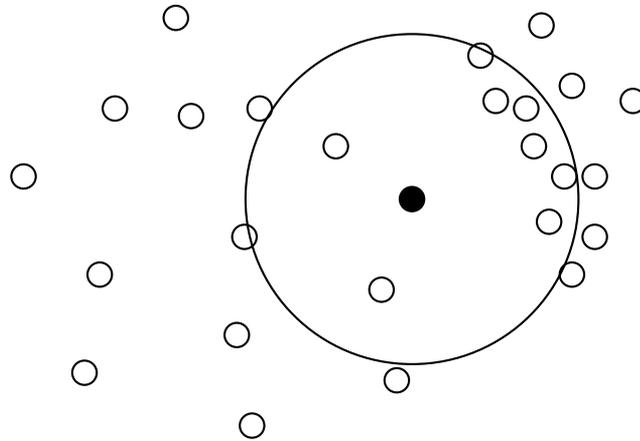


Figure 8.3: Neighbourhood of learning data around test data point based on the Euclidian distance.

Consider the graph in Figure 8.3 where the black dot indicates the test point. If a neighbourhood Euclidian distance, of 8 closest data points to the test point is selected, based on the Euclidian distance, that would result in using the data points inside the large circle.

From the figure one can clearly see that the neighbourhood selected using the Euclidian distance, does not take the data density into account. The density of data points inside the neighbourhood is not uniform and valuable information might be lost. To overcome this problem, one can use the Delaunay tessellations method for selecting the neighbourhood, which is discussed in the next section.

8.4.2 Delaunay tessellations

The disadvantage of the Euclidian distance is that it does not take data density into account, as seen in Figure 8.3. To overcome this problem, determine the Delaunay Tessellation of the data set [18]. The method constructs simplexes among the data points such that no data point is contained in any simplex's circumference. For the two-dimensional data sets, the simplex is a triangle and for three-dimensional data it is a tetrahedron. The resulting Delaunay Tessellation for the example is given in Figure 8.4.

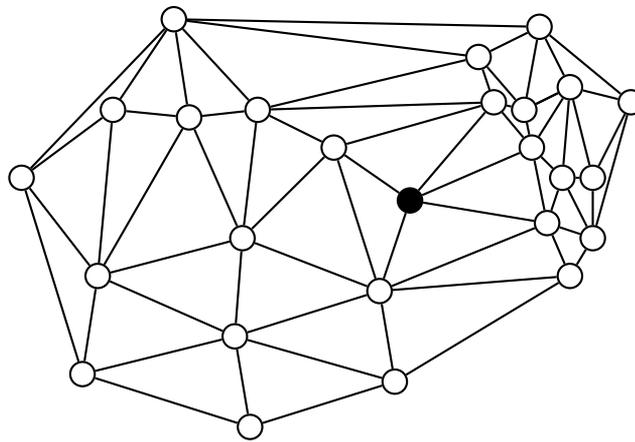


Figure 8.4: Delaunay Tessellations of learning data around test data point.

The neighbourhood of the test point is then determined by the neighbouring simplexes. The immediate neighbouring simplexes are those simplexes that contain the test point as one of their nodes. A first level Delaunay neighbourhood contains the data points of the first layer of simplexes, as seen in Figure 8.5. The first layer of simplexes is indicated by the light gray area and the data points that will form such a neighbourhood are coloured dark gray.

A second level simplex is a simplex that consists of at least one of the nodes of a first level simplex. The data points of first and second layer of simplexes combined, defines a Delaunay neighbourhood of the second level. In Figure 8.6 the simplexes in the two layers are indicated again by the light gray area. The learning data points that form a second layer neighbourhood are coloured dark gray.

Using Delaunay tessellations to construct neighbourhoods, the data density is more uniform in all dimensions. From Figure 8.6 one can see the data density expands equally in all directions as the layers are increased. In high-dimensional data sets (5 dimensions and more), this approach will construct neighbourhoods that represent the data set far better than the neighbourhoods constructed by the Euclidian distance measure.

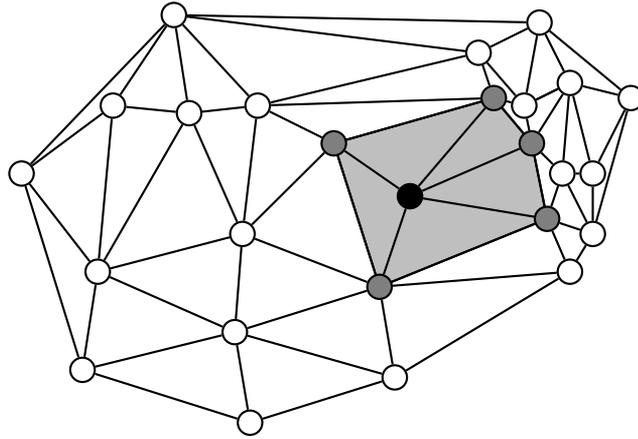


Figure 8.5: Neighbourhood learning data around test data point based on the first level of Delaunay Tesselations.

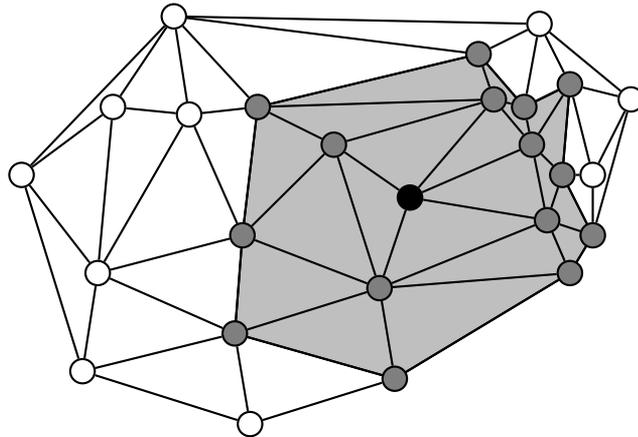


Figure 8.6: Neighbourhood learning data around test data point based on the second level of Delaunay Tesselations.

The parameter to be set in the software tool is level of neighbourhood. In high-dimensional spaces, a level two neighbourhood consists of almost all the data points in the data set.

Algorithm J. Transduction

-
- Step 1:** Select the learning set S the learning set.
- Step 2:** For an observation $\hat{\mathbf{x}}$ determine neighbourhood N of data points from S around $\mathbf{t}\mathbf{x}$:
for given distance measure DM and neighbourhood parameter d ,
 $N_{\hat{\mathbf{x}}} = \{\mathbf{x} \in S | DM(\mathbf{x}, \hat{\mathbf{x}}) < d\}$
- Step 3:** If $N_{\hat{\mathbf{x}}} \notin N$ where $N = \{N_1, N_2, \dots, N_k\}$ the set of all existing neighbourhoods,
- build $SVM_{\hat{\mathbf{x}}}$ based on learning data in $N_{\hat{\mathbf{x}}}$
 - save $SVM_{\hat{\mathbf{x}}}$ model in
 $SVM = \{SVM_1, SVM_2, \dots, SVM_k\} \cup SVM_{\hat{\mathbf{x}}}$
 - save neighbourhood $N_{\mathbf{t}\mathbf{x}}$ in $N = \{N_1, N_2, \dots, N_k\} \cup N_{\hat{\mathbf{x}}}$
- else $SVM_{\hat{\mathbf{x}}} = SVM^*$ where SVM^* is the model corresponding to neighbourhood $N^* = \{N^* \in N | N_{\hat{\mathbf{x}}} = N^*\}$.
- Step 4:** Predict the output \hat{y} of $\hat{\mathbf{x}}$ using $SVM_{\hat{\mathbf{x}}}$.
- Step 5:** Update learning set (if necessary) with test point: $S = S \cup (\hat{\mathbf{x}}_i, \hat{y}_i)$.
- Step 6:** Return to Step 2.
-

8.5 Conclusion

Consider the original design requirement:

The inferential sensor should be able to perform novelty detection and implement a procedure to adapt the model to the changed conditions which were detected as novel information.

Perhaps one of the most important design requirements of the new generation of inferential sensors is its ability to adapt to changing conditions due to dynamic processes. Unless inferential sensors possess the ability to adapt, their widespread use in real-world applications will be very limited. Currently, their lifespan is too short which often results in high maintenance costs.

Before a inferential sensor can become adaptive, it needs to know when to adapt. Therefore a reliable novelty detection step is needed. A new approach for performing novelty detection that uses the characteristics of support vectors was introduced and discussed. The advantage of using this approach is that the novelty is detected independent of the error statistics of the current model.

This chapter also discussed various forms of adaptivity and introduced different levels based on the impact on the operational status of the inferential sensor. These

level vary from low impact adaptation like updating history sets and switching to other known models to a complete offline rebuilding of the inferential sensor.

One specific level of adaptation, the Level 4 Adaptation, uses the transductive modelling approach which was discussed in detail. Issues involving the selection of sets of neighbouring observations to a point of interest were discussed and illustrated. The pseudo-code of the implemented transductive approach in the software package was given.

The research did not fully explore all possible levels and forms of adaptivity that exists. Only an overview of the various scenarios encountered by the engineers of The Dow Chemical Company were considered. A closer inspection of the effects these adaptations have on the process leads to the different levels of adaptation.

The different levels of adaptivity were not implemented fully in the software toolbox, since the toolbox is primarily for model building purposes. Normally, the adaptive part of the inferential sensor would be implemented using rule based or fuzzy logic systems into which the modelling software can be integrated.

It is clear, from our point of view, that only the surface of adaptive inferential sensors has been scratched and much more work needs to be done to fully explore all possibilities.

Chapter 9

Self-Diagnostic Capabilities

9.1 Introduction

In order to survive in a real-life industrial environment, an inferential sensor needs to be able to evaluate its own performance [70]. This requires that more intelligence is built into the inferential sensor so that a process engineer receives some feedback about the accuracy of the predictions made by the current model. This capability is essential in safety-critical applications. For example, it is of utmost importance in the control of chemical processes that use materials that are hazardous to humans and the environment. Therefore, in order to prevent a false sense of trust and limit the risk of spills and accidents, the process engineer needs to be warned when the model detects that its predictions are not reliable anymore [40].

Self-diagnostics is not only used for evaluating a model's performance, it may also be used to evaluate the performance of other sensors [112] or indicate that the process is being run at off-specification operating conditions. Possible faulty equipment could then be inspected and replaced if needed. In the case of off-specification conditions, the process engineer could adjust the process controls to the desired specifications. Such actions will reduce the manufacturing of off-spec products, resulting in a huge cost advantage [40].

To some extent a self-diagnostic capability is connected to novelty detection and adaptation. When novelty is detected and some form of adaptation is required, the inferential sensor has diagnosed itself as not being reliable anymore. There is a slight difference, though. Under self-diagnosis we understand that the inferential sensor gives some kind of confidence level or reliability measure for the prediction it has made. When the inferential sensor is not confident that the current prediction is made with high accuracy, then there is the possibility of a novelty.

For the inferential sensor to diagnose the reliability of its predictions, it is necessary to associate some measure of uncertainty with the model of the learning machine [40]. The lack of the widespread use of SVMs in industry could partly be explained by the absence of confidence intervals associated with the predictions made by the model. The level of confidence in a prediction is often as important as the prediction itself.

One of the main problems encountered in inferential sensor development is how to

determine the uncertainty. In this research we explored three possible ways to obtain uncertainty measures for SVM for regression. The first is a post-learning procedure in which the expected prediction values and their confidence is determined [31]. Another possibility is to construct a model disagreement measure based on the behaviour of several models [83],[41]. Finally, through the application of the Bayesian Framework, error bars can be assigned to the predictions being made.

Aside from uncertainty measures, there are also some other statistical measures that are used to access the overall performance of a inferential sensor. These include error statistics like correlation coefficient and standard deviation, as well as order statistics like the residual analysis and tests for normality of the errors.

This chapter consists of four sections. The first section gives an overview of other statistical measures that are frequently used for assessing the overall performance of the model. Thereafter we briefly discuss three possible approaches for determining uncertainty levels. It is shown in the second section how a post-learning confidence can be obtained for SVMs. In the third section we briefly mention the error bar estimation obtained from applying Bayesian inference to SVMs, which was recently given in [20]. Finally, we explore the possibility of using a non-statistical uncertainty measurement which is used in NN applications, namely the model disagreement measure.

9.2 Error statistics

There are some overall error measures that can be used to assess the performance of the model. These measures, commonly known as error statistics, can be used for any type of learning machine [71],[43]. The error statistics evaluate the relations between observed output y and the predicted output \hat{y} . It is assumed that both variables are random variables and the variances and covariances exist for both. Furthermore, their variances need to be nonzero.

Standard deviation

The standard deviation gives an indication of how spread out the values of the sample are about its mean. Let us define first two statistical measures: the mean and the variance. The mean of a sample \bar{t} is the sample's average value and is determined as

$$\bar{t} = \frac{\sum_{i=1}^{\ell} t_i}{\ell},$$

where ℓ is the number of observations in the sample. The mean is also known as the expectation of t and denoted by $E[t]$. The variance of the sample measures the spread or dispersion about the mean of the sample values and is given by

$$\sigma^2(t) = \frac{\sum_{i=1}^{\ell} (t_i - \bar{t})^2}{\ell - 1}.$$

In some literature σ^2 is also denoted with $\text{Var}(t)$.

Now we can define the standard deviation $\sigma(t)$ as the square root of the variance. For the observed output y and the predicted output \hat{y} , we determine the standard

deviation of the difference $(y - \hat{y})$, i.e.,

$$\sigma(y - \hat{y}) = \sqrt{\text{Var}(y - \hat{y})}. \quad (9.1)$$

Correlation coefficient

The correlation coefficient is a dimensionless quantity between zero and one, which measures the association between two variables. For the observed y and predicted \hat{y} , the correlation coefficient is determined as

$$c = \frac{\text{Cov}(y, \hat{y})}{\sqrt{\text{Var}(y)\text{Var}(\hat{y})}}, \quad (9.2)$$

where the covariance of y and \hat{y} is determined by

$$\text{Cov}(y, \hat{y}) = \frac{\sum_{i=1}^{\ell} (y_i - \bar{y}) \sum_{i=1}^{\ell} (\hat{y}_i - \bar{\hat{y}})}{\ell}.$$

If the correlation coefficient of two variables is close to one, there is a strong relation between the observed output y and the predicted output \hat{y} . It is an indication that the model is able to reproduce the observed output.

R^2 -statistics

The R^2 -statistics or coefficient of determination is a quantity that gives an indication of the proportion of variance explained by the model [56]. It is defined as

$$R^2 = 1 - \frac{SS_E}{S_{yy}},$$

where SS_E is the sum square error

$$SS_E = \sum_{i=1}^{\ell} (y_i - \hat{y}_i)^2,$$

and S_{yy} is determined as

$$S_{yy} = \sum_{i=1}^{\ell} (y_i)^2 - \ell \bar{y}^2.$$

The sum square error measures the variability in y remaining after the model has been considered and S_{yy} measures the variability in y without considering the effect of the model. Therefore,

$$R^2 = 1 - \frac{\sum_{i=1}^{\ell} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{\ell} (y_i)^2 - \ell \bar{y}^2}, \quad (9.3)$$

gives the proportion of variation explained by the model.

Because $0 \leq SS_E \leq 1$, it follows that $0 \leq R^2 \leq 1$. If R^2 is close to one it is an indication that most of the variability in y is explained by the model. However, one should not infer from large R^2 values that the model is an accurate model [56].

Relative error

Consider

$$\sum_{i=1}^{\ell} (y_i)^2 - n\bar{y}^2 = n \left(\frac{\sum_{i=1}^{\ell} (y_i)^2}{n} - \bar{y}^2 \right) = n\sigma^2(y),$$

and

$$\frac{SS_E}{S_{yy}} = \frac{\left(\sum_{i=1}^{\ell} (y_i - \hat{y}_i)^2 \right) / n}{\sigma^2(y)}.$$

Now taking the square root over both sides, gives

$$\sqrt{\frac{SS_E}{S_{yy}}} = \frac{\sqrt{\left(\sum_{i=1}^{\ell} (y_i - \hat{y}_i)^2 \right) / n}}{\sigma(y)}, \quad (9.4)$$

which is often referred to as the relative error. Let us denote the relative error with RE . Then the R^2 -statistic can also be written as

$$R^2 = 1 - (RE)^2.$$

Root mean square error of prediction(RMSEP)

The mean square error is defined as

$$MSE = \frac{\sum_{i=1}^{\ell} (y_i - \hat{y}_i)^2}{n},$$

and gives the expected squared deviation of the observed y and their predictions \hat{y} . It is therefore a measure of the size of the measurement error.

The root mean square error is the square root of the MSE and has the same units as the observed y . The RMSEP for the observed y and predicted \hat{y} is given by

$$RMSEP = \sqrt{\frac{\sum_{i=1}^{\ell} (y_i - \hat{y}_i)^2}{n}} = \sqrt{\frac{SS_E}{n}}. \quad (9.5)$$

Thus, the $RMSEP$ measures the average variability in y that remains after the model has been considered. The $RMSEP$ can also be determined using the relative error and the standard deviation: $RMSEP = E_r * s(y)$.

Vapnik's measure

Another overall error measure that can be used it is Vapnik's measure [11] and is often used as a model selection criterion. This error measure uses the estimation of the upper bound of the prediction risk provided by SLT and is given by

$$\text{Prediction Risk} \leq \frac{R_{\text{emp}}}{(1 - c\sqrt{\varepsilon})_+}, \quad (9.6)$$

where

$$\varepsilon = 4 \frac{h_n(\ln(2\ell/h_n) + 1) - \ln(\eta/4)}{\ell}.$$

Here h_n is the VC-dimension of the set of loss functions and c is a constant that reflects the “tails of the loss function distribution”. Recall that in Chapter 2 we explained that the “tails of the distribution” give the probability of observing large values of loss. Furthermore, the upper bound in (9.6) holds true with probability $1 - \eta$. For practical purposes $c = 1$ and $\eta = \min(4/\sqrt{n}, 1)$ are recommended by [108],[11]. An estimate of the VC-dimension of a given learning machine model that Vapnik proposed [106] is:

$$h_{est} = \min(\ell, R^2 \| \mathbf{w}_0 \|^2) + 1, \quad (9.7)$$

where

$$\| \mathbf{w}_0 \|^2 = \boldsymbol{\omega}^T K(\mathbf{x}, \mathbf{x}) \boldsymbol{\omega},$$

and R is the radius of the smallest sphere containing all the training data and it can be estimated using

$$R^2 = \min_a \max_{\mathbf{x}_i} (K(\mathbf{x}_i, \mathbf{x}_i) + K(a, a) - 2K(\mathbf{x}_i, a)).$$

Finally, R_{emp} depends on the loss function used. For linear ϵ -insensitive loss

$$R_{emp} = \frac{1}{\ell} \sum_{i=1}^{\ell} (|y_i - \hat{y}_i| - \epsilon),$$

and for quadratic loss

$$R_{emp} = \frac{1}{\ell} \sum_{i=1}^{\ell} (|y_i - \hat{y}_i| - \epsilon)^2.$$

When comparing different models, the model with the smallest value of Vapnik’s measure is selected as the model with the lowest prediction risk.

Error distributions and residual information

The difference between the observed y and the predicted \hat{y} , is called the residual and is denoted by

$$r_i = y_i - \hat{y}_i.$$

Since a residual may be viewed as the deviation of the observed output and its prediction, it measures the variability not explained by the model. One can also think of residuals as the observed values of the errors. Therefore, any departures from the underlying assumptions on the errors should show up in the residuals. Note that residuals have zero mean and their approximate average variance is given by the mean squared error. In order to make residuals comparable they are often standardised such that

$$\tilde{r}_i = \frac{r_i}{\sqrt{MSE}}.$$

The standardised residuals have zero mean and unit variance.

Various types of residual plots can be considered [56]. In the software tool, we consider only three types. The first is a histogram plot of the (standardised) residuals' distribution. This graph is used to view the error frequency. Since it is assumed that the errors are normally distributed, the shape of the histogram plot should roughly follow the normal density function.

The second graph used, is the plot of (standardised) residuals against the predicted \hat{y} . These graphs are useful for detecting several common types of model inadequacies. Ideally the residual plot should show no patterns but be contained in a horizontal band. If for example a funnel pattern emerges, it is an indication that the variance of the error is not constant. A curved residual plot indicates nonlinearity that is not captured by the model. The residual plot against \hat{y} may also reveal unusually large errors. These observations may be considered as outliers. However, large errors at the extreme of \hat{y} could indicate that either the variance is not constant or the model does not capture the true relationship between the inputs and the outputs.

The third graph being used is the plot of (standardised) residual against \mathbf{x}_j with $j = 1, \dots, n$ and n the number of dimensions. These graphs show the residuals against the input dimension and should show no relation to the input dimensions. Thus, as in the previous residual plot, the plot should be contained in a horizontal bar.

For more information on interpreting residual plots, the reader is referred to standard statistical literature like [56],[71].

Normal probability plots

A normal probability plot of the residuals is used to see whether there is a gross departure from normality. It is often assumed that the errors or residuals are normally distributed with zero mean and an approximate average variance of MSE . Small departures from normality do not effect the model greatly, but gross non-normality is potentially more serious as the confidence and prediction intervals (like the error bars) depend on the normality assumption.

The normal probability plot is constructed as follows.

1. Let $e_{[1]} < e_{[2]} < \dots < e_{[\ell]}$ be the (standardised) residuals ranked in increasing order.
2. Approximate the expected normal value, $E[e_{[i]}]$ with

$$\Phi^{-1} = (i - \frac{1}{2})/\ell, \quad i = 1, \dots, \ell,$$

where Φ denotes the standard normal cumulative distribution.

3. Plot the probabilities $E[e_{[i]}], i = 1, \dots, \ell$ against the errors $e_{[i], i=1, \dots, \ell}$.

If the errors are indeed normally distributed, the normal probability plot should be a straight line. Heavy-tailed distributions show at the extremes sharp departures from the straight line. The error distribution is then considered not to be normal. Light-tailed distributions tend to flatten at the extremes. Other patterns may also emerge. Standard statistical literature discusses these in detail [56],[71].

Furthermore, the occurrence of one or two large residuals will show up on the normal probability plot. Therefore, normality plots are often used to check for the presence of outliers.

9.3 Classical confidence

The way classical confidence is often obtained from data driven models is to assume that the learning machine model is a perfect model of the problem in question. Note that this is a big assumption to make and it is in general not true. However, in many industrial applications this approach for determining confidence is used and therefore needs some consideration. In this section we will show the disadvantages of this approach.

Under the assumption that the learning machine model is perfect, the predicted output values and their corresponding observed output values drawn from the (independent) test set must be identical. The plot of the observed values against the predicted values must form a straight line with gradient one and intercept zero.

As there is a close relationship between correlation analysis and fitting straight lines by the least squares method, we first consider the correlation coefficient between the predicted and observed output values [71]. For a perfect model, the correlation coefficient should approach one since there is a perfect correlation between such a model and the data. Of course, we know that this will seldom be the case and there will always be an error between the predicted output of the SVM model and the observed output [31]. As a result, the correlation coefficient will deviate from one. Therefore, it is necessary to test for statistical significance of the correlation coefficient. The statistical significance of the correlation coefficient is tested using a one-sided t-test. If the test shows that there is no correlation between the observed and predicted output, the SVM model should be changed.

Now, in case of a significant correlation coefficient, we can proceed to find the best least squares fit for the straight line. Let $\hat{y}_i, i = 1, \dots, \ell$, be the predicted values of the observed y_i . The obtained regression line is then used to modify the SVM model's prediction \hat{y}_i such that

$$\tilde{y}_i = m \cdot \hat{y}_i + c, \quad (9.8)$$

where c and m are the intercept and slope of the regression line respectively. \tilde{y}_i is now the estimate of the true value. The slope of the regression line should be close to one, otherwise the SVM model is inconsistent with the data. In practice, the SVM model must be rebuilt until the condition is met.

For the model in (9.8) we can provide confidence limits around the mean response of (9.8), if we assume that the errors around the regression line are normally distributed [56],[31]. The $100(1 - \alpha)\%$ confidence interval on the mean of the response of (9.8) at the point \hat{y}_i , is given by

$$I = \tilde{y}_i \pm t_{\alpha/2, \ell-2} \sqrt{\frac{\sum_{j=1}^{\ell} (y_j - \tilde{y}_j)^2}{\ell - 2} \left(\frac{1}{\ell} + \frac{(\hat{y}_i - \bar{\hat{y}})^2}{\sum_{j=1}^{\ell} (\hat{y}_j - \bar{\hat{y}})^2} \right)}, \quad (9.9)$$

where \tilde{y}_i is the expected value determined by (9.8), $t_{\alpha/2, \ell-2}$ the t statistic with $\ell - 2$ degrees of freedom.

The confidence interval in (9.9) is not appropriate for determining the confidence of a new observation, since it is not a probability statement about future observations but an interval estimate of the mean of y [56]. Therefore, we need to determine the prediction interval for a future observation y_0 . The $100(1 - \alpha)$ % prediction interval on a future prediction \hat{y}_0 is:

$$I = \tilde{y}_0 \pm t_{\alpha/2, \ell-2} \sqrt{\frac{\sum_{j=1}^{\ell} (y_j - \tilde{y}_j)^2}{\ell - 2} \left(1 + \frac{1}{\ell} + \frac{(\hat{y}_0 - \bar{\hat{y}})^2}{\sum_{j=1}^{\ell} (\hat{y}_j - \bar{\hat{y}})^2} \right)}. \quad (9.10)$$

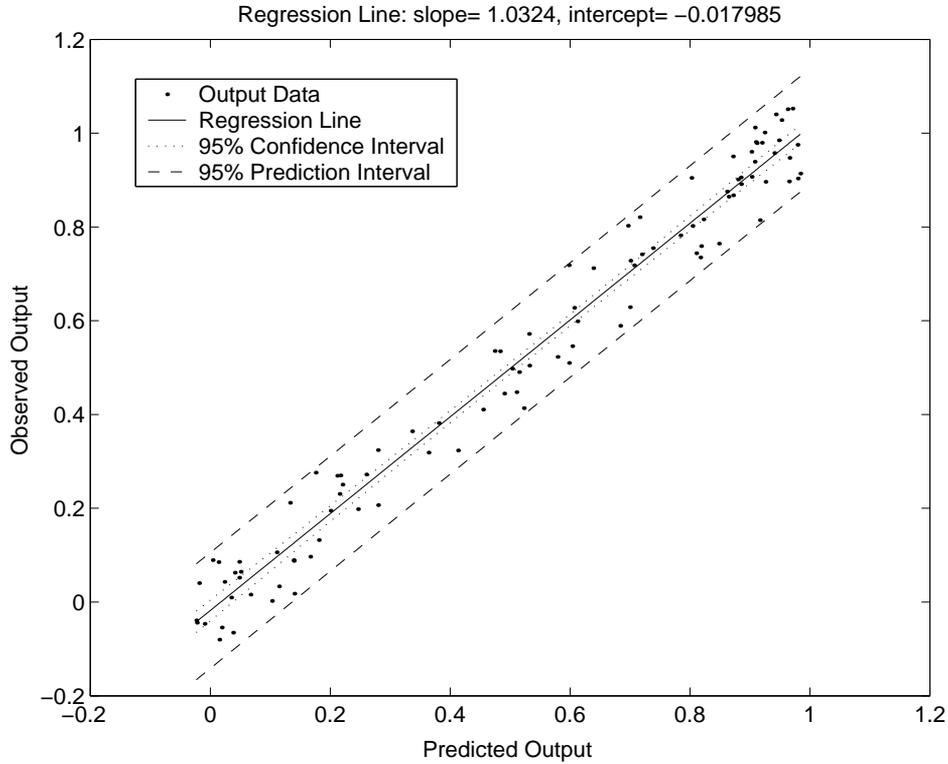


Figure 9.1: Confidence and Prediction Interval for Example 1 in Chapter 2.

In Figure 9.1 the observed output is plotted against the predicted output. The determined regression line with the 95% confidence and prediction intervals can also be seen. Since the slope of the regression line is very close to one, the SVM model is almost perfect and therefore the confidence interval is very narrow. Note that the confidence interval is in effect a pair of parabolic lines with the narrowest interval at the point where the bulk of the data lies as seen in Figure 9.2. In fact, the minimum

in the confidence interval occurs at the mean of the x -axis and y -axis data. This type of confidence is therefore only a reflection of the mean of the data and does not give the true uncertainty associated with a prediction. Another form of uncertainty measurement is therefore needed.

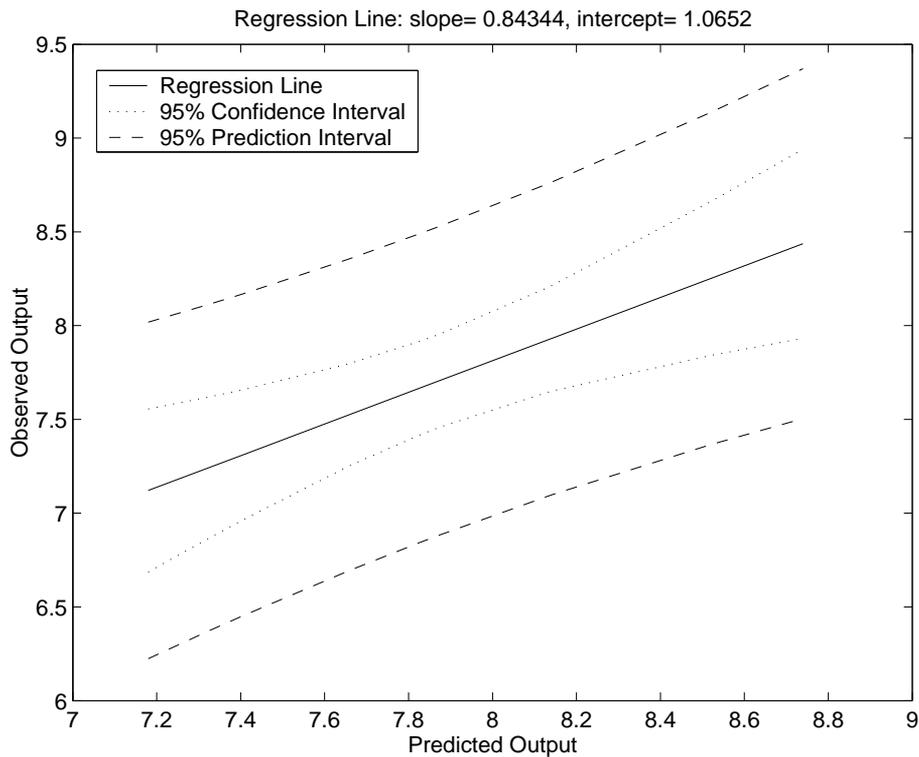


Figure 9.2: Confidence and Prediction Interval around the regression line.

9.4 Error bar estimation

Another statistical approach to associate uncertainty with a prediction is the error bar estimation. Error bars naturally arise from applying Bayesian Techniques to a learning machine. A Bayesian interpretation of the Support Vector Machine has first been posed in [93] where it has been shown that the use of different kernels in SVMs can be viewed as defining different prior probability distributions on the function space. However, incorporating the popular Bayesian techniques with priors on the function space cannot be done as easily as for priors on the weight space, like in [50]. Recently, the Bayesian Framework was extended to SVM for regression [20],[12],[44],[16].

In the Bayesian Framework three probability terms have to be determined, namely

the *prior* probability, the likelihood and the evidence, in order to determine the *posterior* probability [63]. The prior probability $P[\mathbf{f}(X)]$ of the unknown function $f(\mathbf{x})$ at X embodies a *priori* knowledge of the function. It is used to impose constraints on the model such that significant probability is assigned only to those functions that satisfy the constraints. The likelihood is the probability $P[D|\mathbf{f}(X)]$ that, if $f(\mathbf{x})$ is the underlying function of the data D , by random sampling the function $f(\mathbf{x})$ at X will produce Y . It can also be considered as the noise distribution of the additive model $y_i = f(\mathbf{x}_i) + \delta_i$ if δ_i is i.i.d. with probability distribution $P[\delta_i]$. The evidence $P[D]$ is the likelihood of the data for a given model. Finally, the posterior probability of the function $\mathbf{f}(X)$ is determined by using Bayes' Rule ¹,

$$P[\mathbf{f}(X)|D] = \frac{P[D|\mathbf{f}(X)]P[\mathbf{f}(X)]}{P[D]}.$$

Error bars for a prediction can be computed if the mean and variance of a given posterior distribution is known. The derivation of the probability functions required can be found in [20],[44] and [94]. As the formula for the estimation of the error bar for the ϵ -SVM was published in 2002 in [20], this measure of uncertainty has not been tested extensively in simulations.

9.5 Model disagreement

Since the confidence interval for many learning methods is based on the mean of the data and not taking the density of the learning space into account, the confidence interval often does not reflect the effect of varying data density [82].

The idea is to consider different models of similar complexity with the freedom to make mistakes in different parts of the input space as a function of the local density [83]. In areas where enough data are present the different models should lie in close proximity of each other, whereas in areas where less data are available the models should have more freedom. The confidence of a selected model should be large in that area where the other models display the same behaviour. However, in areas where the data density is low, different models behave differently with the result that the confidence in a selected model should be lower.

It has been shown in industrial applications of NNs that the use of a collection of networks gives more robust models that include confidence limits based on the standard deviation of stacked neural nets [41]. This notion of confidence, is called *model disagreement*. In areas where different models disagree with each other, there is high level of disagreement among the models and therefore the value of the measurement of disagreement is high. Similarly, in areas where different models agree with each other, there is a low level of disagreement, resulting in a small value for the measurement of disagreement.

One of the main issues in this approach is how to construct various models which are sufficiently different from each other but have the same level of complexity. If comparable complexity was not required one would try to compare apples with oranges,

¹For events A and B , the probability of event A given event B , is $P[A|B] = \frac{P[B|A]P[A]}{P[B]}$.

since the models would not only differ in their error term but also in their complexity. In the case of NN applications, constructing different models of the same complexity can be done by keeping the number of hidden nodes fixed.

Note that no assumptions on the prior or posterior probability distributions are made, nor is normality of the errors assumed. The only assumption being made is that the set of models used are not highly correlated and therefore their modelling errors are not correlated.

Ideally, the model disagreement measure should be extended to Support Vector Regression. However, it is not clear how to construct different models which vary in their error terms but not in their complexity terms. There are basically four parameter choices that could be used: ϵ , C , the kernel and the kernel parameter. In support vector regression, the error term is mainly controlled by the size of ϵ . However, ϵ also controls the number of support vectors and therefore the complexity so it cannot be varied. Since the regularisation parameter C controls the trade-off between the error term and complexity term, it too must remain fixed to assure the same level of trade-off for different models. What remains are the kernel choice and the kernel parameter. Since the type of kernel function greatly influences the behaviour of the resulting SVM model, it is a possible candidate to consider. The application of the model disagreement measure to SVMs for regression may be an interesting new field of research.

9.6 Conclusion

Let us state the design requirement again:

The requirement for the inferential sensor is that it should have self-diagnostic capabilities in order to evaluate its own reliability.

In this chapter we considered in addition to the error statistics that measure the overall performance of the inferential sensor also three possible ways to assign an uncertainty measure with a prediction made by the inferential sensor. These measures are necessary to evaluate the reliability of the inferential sensor.

The error statistics that are used for evaluating the overall performance of a model include the standard deviation, the correlation coefficient, the relative error and root mean square error prediction. An error measure originating from SLT, namely Vapnik's measure is also given. Residual plots are used to verify that the errors contain no patterns, which is an indication that the model explains the variance in the data. Furthermore, since it is assumed that the error between the observed output y and the predicted output \hat{y} is normally distributed, the normal probability plot of the error is used to test for normality.

For learning machines, like support vector machines, which are derived without prior knowledge or assumptions on the underlying distribution, classical confidence limits can only be constructed in a post-learning way. Such an uncertainty measure actually gives only confidence about the mean of the data and does not take data density into account. It is therefore recommended to use this approach only when no other measure of uncertainty is available.

Very recently it has been shown that through applying Bayesian techniques, an uncertainty measure in the form of an error bar can be derived for support vector machines for regression. Error bars assign an uncertainty level to a prediction and can therefore be used for assessing the accuracy of a prediction made by the inferential sensor as well as monitoring the inferential sensor's performance. However, no extensive practical results are available yet on the industrial implementation of this approach.

In industrial applications of NN and GP's it has been shown that the model disagreement measure provides a practical alternative which takes data density into account and does not make any assumption on the prior distribution. However, much more research is needed in order to extend this approach to SVMs, since the approach requires the construction of various model of comparable complexity but different errors in areas of low data density. In our point of view, this may be a new and interesting field of research.

The new advances in support vector machine like the error bar estimation and the model disagreement measure hopefully will enable the inferential sensor to associate with its prediction a level of uncertainty. In addition, the error statistics give the inferential sensor the ability to assess its overall performance. Combining both measures gives the inferential sensor the ability to monitor its performance and evaluate its reliability.

Part IV

Implementation

Chapter 10

Implementation of Software

10.1 Introduction

In this final part of the thesis we show examples of the incorporation and implementation of the identified design requirements into software to build support vector machines. The software is intended for fast prototyping purposes only, but has been successfully used in industrial applications within The Dow Chemical Company [41].

The software was built using the MATLAB programming language [54]. It has been observed that the MATLAB internal QP tool may not be optimal and therefore we opted for an external off-the-shelf optimising tool. Note also that for online applications where speed is essential other programming languages like C++ will be a better choice.

The algorithmic code presented throughout the thesis represents the core of the software. In order to make the software more user friendly, a user interface has been built. The user interface also safeguards the parameters in order to prevent run time errors that are most disturbing for the user. The tool also includes procedures for viewing the results, examining the error statistics and exporting figures and data for documentation.

The software uses a data structure in which data and parameters are stored in fields. Since data are received in many different forms (Excel files, ascii files, text file) a data preprocessing routine is included. The data preprocessing not only constructs valid data structures of the training and testing data but also allows the user to supply names to the input and output variables. These names are used to construct result figures with informative titles and axes labels.

The software consist of two user interface windows. The first window, as seen in Figure 10.1, allows the user to select the type of problem, the application and two basic parameters, namely the kernel and the complexity parameters. This window also has three menu bars, namely File, Execute and Results. In the file menu, the user can either select an existing data file or construct one using the data preprocessing tool. The second menu contains execution commands. The execution demands depend on the application being used and whether any optimisation options are used. The third menu, named Results, allows the user to display the obtained modelling results and

save the results in a data file.

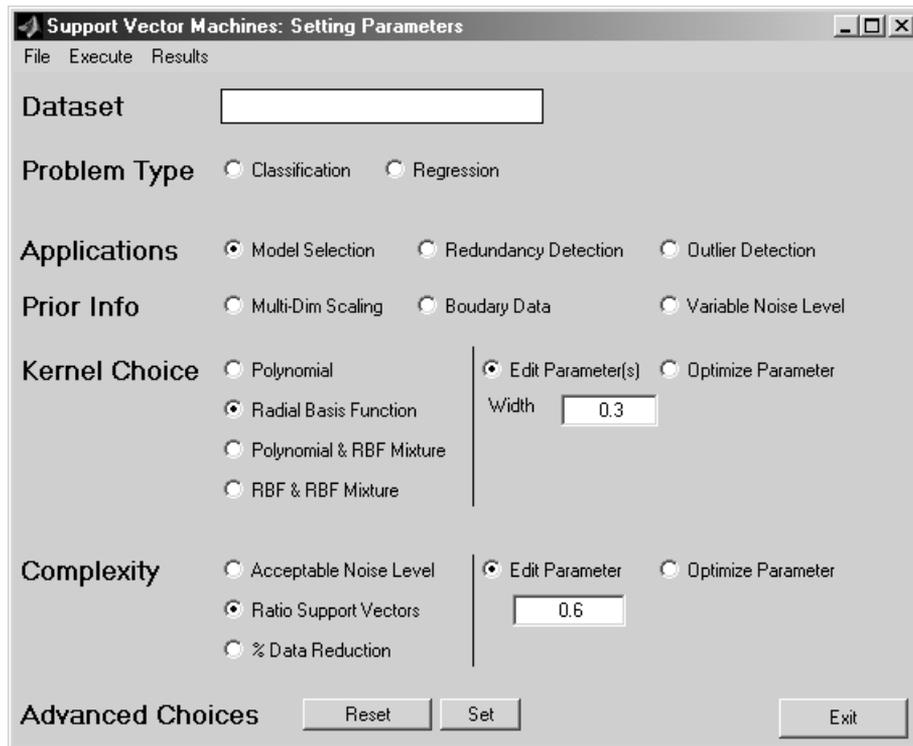


Figure 10.1: User interface for parameter setting.

The second user interface window can be viewed by using the option Advanced Choices in Figure 10.1. It contains more advanced parameter settings such as the type of optimisation method used and the type of loss-function used. In Figure 10.2 all the options available can be seen.

Although the software contains the implementation of SVMs the development of inferential sensors is not limited to the use of SVMs. From our point of view the development of inferential sensors requires the combined use of SVMs with other techniques like NNs and GP. Each learning machine has its advantages and disadvantages. One should use an appropriate technique for the problem at hand. Such an example can be found in [41].

In the rest of the chapter, we show examples of the implemented software. In the first section, the use of the support vector machine and the choice of its parameters as discussed in Part I are illustrated. The second section contains examples of implementations of Part II, which include the outlier and redundancy detection as well as multi-dimensional scaling. The implementation of elements of Part II is shown in the third section. Finally, we discuss in short some computation issues involving the implementation of support vector machines.

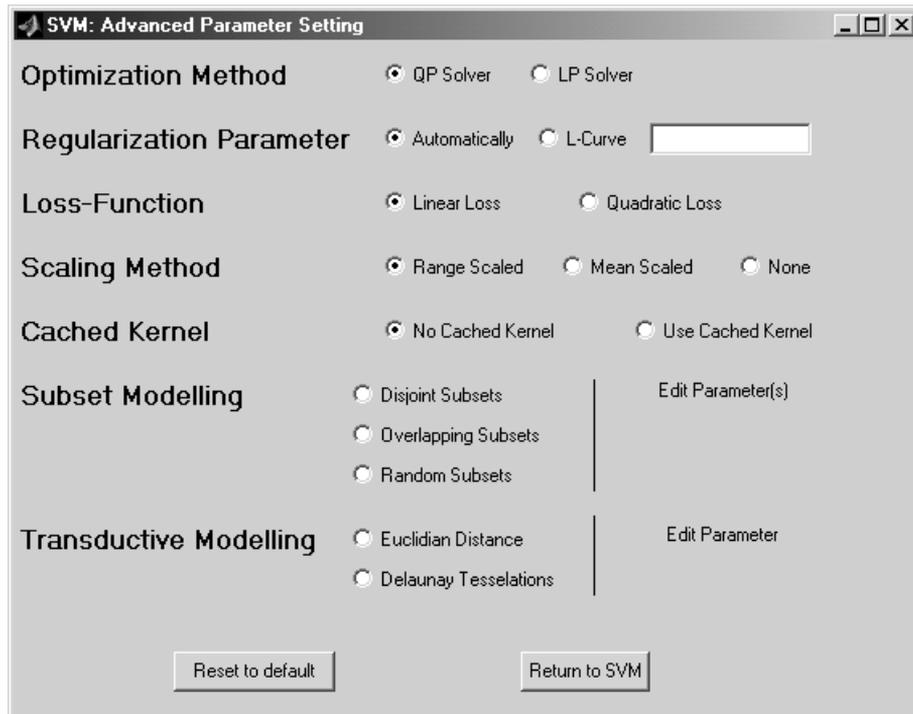


Figure 10.2: User interface for advanced parameter settings.

10.2 Applications of Part I

10.2.1 Complexity control

In this part the SVM method for regression is implemented. The optimisation of the ϵ and ν parameters is an additional component.

Implementation of SVM

Both classification and regression type of problems can be solved using this software. The user makes a choice between a classification or regression problem type. Algorithmic code A is implemented for the SVM for classification. The implementation of the ϵ and ν -SVM involves the use of the Algorithmic codes B and C respectively. The implementation of both codes are such that the type of loss function used is merely a parameter choice. Note that since the classification SVM requires no complexity control the options for the complexity control parameters are removed.

Setting of the values for ϵ and ν is done in the interface window of Figure 10.1 depending on which type of SVM regression (ϵ -SVM or ν -SVM) is selected. Note that the user interface safeguards the values entered for the complexity parameters such that $\epsilon \leq 0$ and $0 < \nu \leq 1$ are satisfied.

Examples of the ϵ -SVM and the ν -SVM were given in Chapter 2.

Optimising the value of ϵ and ν

The optimisation of the complexity parameters is a separate application. Once the user selects this option, an executing command is available in the Execute menu of Figure 10.1. The user is also required to enter iteration parameters before the application can be executed. The optimisation of the complexity parameter largely involves the use of Algorithm D.

Our implementation, however, is more interactive. It allows the user to inspect the various models constructed during optimisation including their error statistics. The user may also choose to select another value instead of the optimised value.

Consider the data of Example 1 in Chapter 2. The following parameters are set $C = 1000$, linear loss, range scaling and an RBF kernel with $\sigma = 0.3$. The optimisation of ϵ for iteration parameters $\epsilon_start = 0$, $\epsilon_end = 0.5$ and $it_num = 11$ is displayed in Figure 10.3.

Figure 10.3 shows a number of error statistics of the various models plotted against increasing percentages of support vectors. The error statistics corresponding to the optimal value are indicated by the circle. Above the error plots, there is a control bar with which the user can move among the different models according to the percentages of support vectors. The circle moves accordingly. To the left of the control bar the value ϵ corresponding to the model of the encircled error statistics is displayed. Figure 10.3 also has a menu bar with the following options.

- *Display optimal parameter*
This option displays the value of the (selected) optimal complexity parameter. In the case of the ν -SVM this option also displays the evaluated value of ϵ corresponding to ν .
- *Plot best SVM*
The predictions made by the model of the (selected) optimal complexity parameter is plotted in a separate figure window. In this separate window all the performance monitoring options (see Section 10.5) are available.
- *Reset to optimal*
If the user chooses to use the value originally determined by the optimisation routine after all, this options resets the changes that have been made.
- *Save all results*
This option saves the result from the optimisation routine to a data file.
- *Return optimal parameter*
If the user is satisfied with the selected or optimal value, the value can be returned as a setting for Figure 10.1.

The optimisation routine also displays a figure with the predictions of all the models. Inspection of this figure can give the user an idea of the differences in behaviour of the different models.

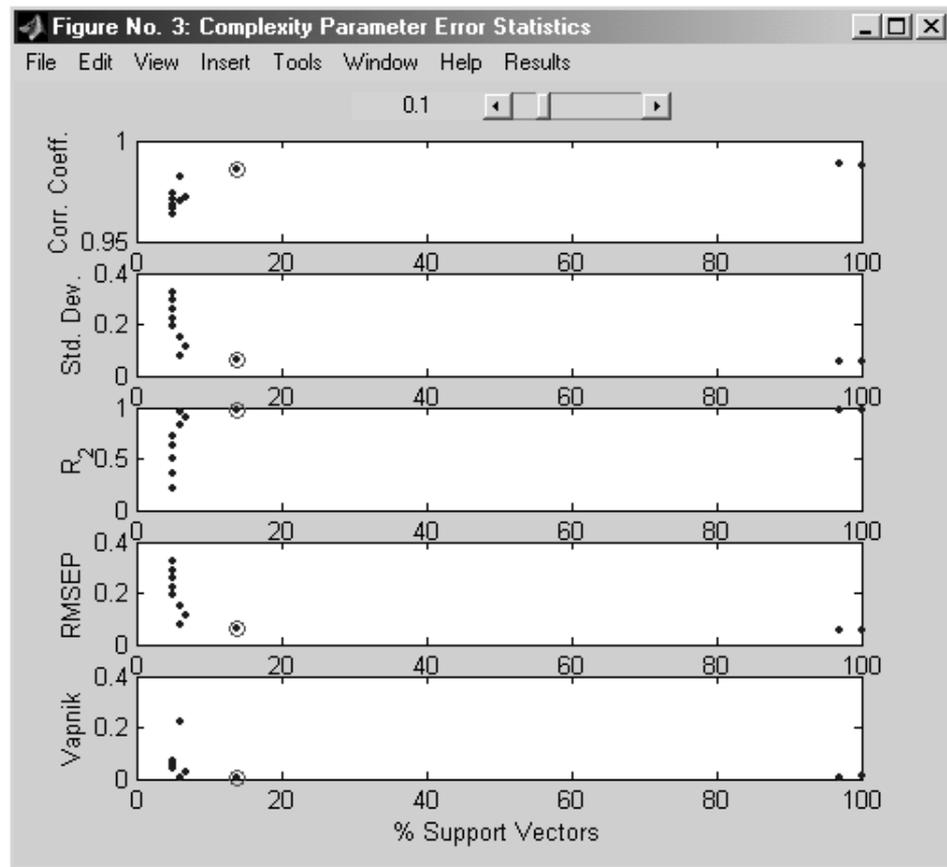


Figure 10.3: Interactive figure for optimising the complexity parameter.

10.2.2 Handling high-dimensional data

The implementations of this chapter comes down to the implementation of various kernels and the optimisation of their parameters.

Implementation of various kernel functions

The type of kernel function is selected in Figure 10.1. Although the user interface only supports four types of kernel functions, many other kernels like β -spline, ANOVA and Fourier kernels are supported by the software. Examples of the RBF and polynomial kernels can be found in Chapter 3.

Upon selecting a kernel function, the user interface assists the user in the type and number of kernel parameters to be set. Again the user interface allows only valid parameter entries. When invalid values are entered, the user receives information about the nature of the error.

Optimisation of the kernel parameter

The optimisation of the kernel parameters is again a separate application. Once the user selects this option, an executing command is available in the Execute menu of Figure 10.1. The user is also required to enter iteration parameters before the application can be executed. The optimisation of the kernel parameter involves largely the use of Algorithm E.

Our implementation is yet again more interactive. It allows the user to inspect the various models constructed during optimisation including their error statistics. The user may also choose to select another value instead of the optimised value.

Consider the data of Example 1 in Chapter 2. The following parameters are set $C = 1000$, linear loss, range scaling and $\epsilon = 0.1$. The optimisation of an RBF kernel for iteration parameters $\sigma_{start} = 0.05$, $\sigma_{end} = 0.55$ and $it_{num} = 11$ is shown in Figure 10.4. Figure 10.4 also shows a number of error statistics of the various models. The error statistics corresponding to the optimal value are indicated by the circle. Note that the subplot in Figure 10.4 contains rankings of the various models which are based on Vapnik's measure. The figure also contains two additional menu bars, namely Add/Remove SVM and Results. In the Results menu bar the following options are available.

- *Choose other optimal parameter*
This option allows the user to select an alternative model in the ranking plot.
- *Display optimal parameter*
Here the kernel parameters corresponding to the indicated index are displayed.
- *Plot best SVM*
The model corresponding to the selected or optimised parameter is displayed in a separate figure. This figure enables all the performance monitoring and features that are available (see Section 10.5).
- *Save results*
The results from the optimisation routine can be saved in a data file.
- *Return optimal parameter*
When the user is satisfied with the parameter optimised or selected, its value can be returned to the user interface.

Another figure showing the performance of all constructed models is also given. Inspection of this figure can give the user an idea of the differences in behaviour of the different models. This figure might get too crowded for analysis, therefore in Figure 10.4 the menu bar Add/Remove SVM was added. This menu bar allows the user to select a number of indexes in the ranking plot of Figure 10.4. The corresponding models are then removed for the performance figure or added again if they were removed previously. Furthermore, the menu bar has an option to reset all changes that have been made.

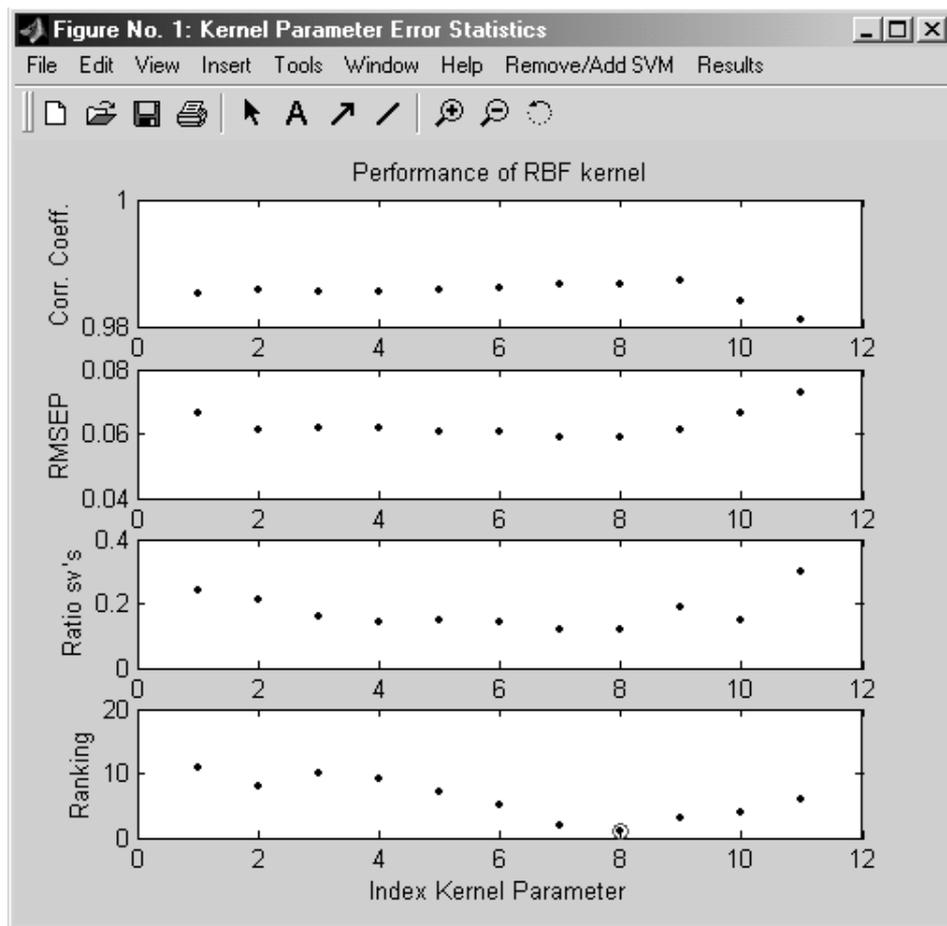


Figure 10.4: Interactive figure for optimising the kernel parameter.

10.2.3 Robustness

The robustness of the SVM for regression is determined by two parameters. The loss function and the regularisation parameter. Note that the implementations of the ϵ -SVM and ν -SVM consider the type of loss function as a parameter choice. Therefore the user only needs to select either linear loss or quadratic loss (see Figure 10.2).

The regularisation parameter is also set in Figure 10.2. Three possibilities are available. One can simply set the value or choose to use the estimation method proposed in Chapter 4. Or one can choose to optimise the value using the L-curve method.

10.3 Applications of Part II

10.3.1 Generalisation ability

The implementation of the content in this chapter only involves the use of the mixed kernel function. The mixed kernel is implemented as an option of the type of kernel. Setting or optimising the kernel parameters of the mixed kernel is therefore no different than for the other kernel functions. Consequently no further explanation is needed.

10.3.2 Data compression and outlier detection

Implementation of the applications of the data compression basically consists of Algorithm H for outlier detection and Algorithm I for redundancy detection. As many of the applications, our implementation is interactive and allows a considerable amount of control for the user. The applications are very similar, therefore we give an example of the outlier detection only. We further give the results of only one iteration.

Our implementation of the outlier detection approach gives a number of figures and allows the user to adjust the cut-off level. In Figure 10.5 the frequency plots as explained in Chapter 6 are shown. The cut-off level which is automatically determined by the approach is displayed as well. Note that the user can adjust the cut-off level using the control button. Adjusting the cut-off level also affects the identified outliers shown in Figure 10.6. Two other figures are also shown. One contains graphs that show the Lagrange multiplier values for each model as seen in Figure 10.7. The other figure displays the error statistics. An example of that is given in Figure 10.8.

Based on the cut-off level and the standard deviation of the RMSEP, which is displayed in Figure 10.6, the user can decide whether the identified indices should be returned and whether a next iteration should be made.

10.3.3 Incorporating prior knowledge

In this chapter we discussed three possible ways of incorporating prior knowledge.

- *Boundary information*

The incorporation of boundary information is basically done by adding artificially generated data points to the data set to reflect the boundary information. The user interface obtains this boundary information from a data file containing the corresponding input and output data of the boundary data points. In order to avoid run time errors, the dimensions of the boundary data are checked beforehand. Chapter 7 contains an example of the use of this type of prior knowledge.

- *Multi-dimensional scaling*

The second possibility was to use sensitivity analysis information for multi-dimensional scaling purposes. In the user interface the sensitivity analysis information is entered per input dimension as seen in Figure 10.9. An example of a multi-dimensional scaling application can be found in Chapter 7.

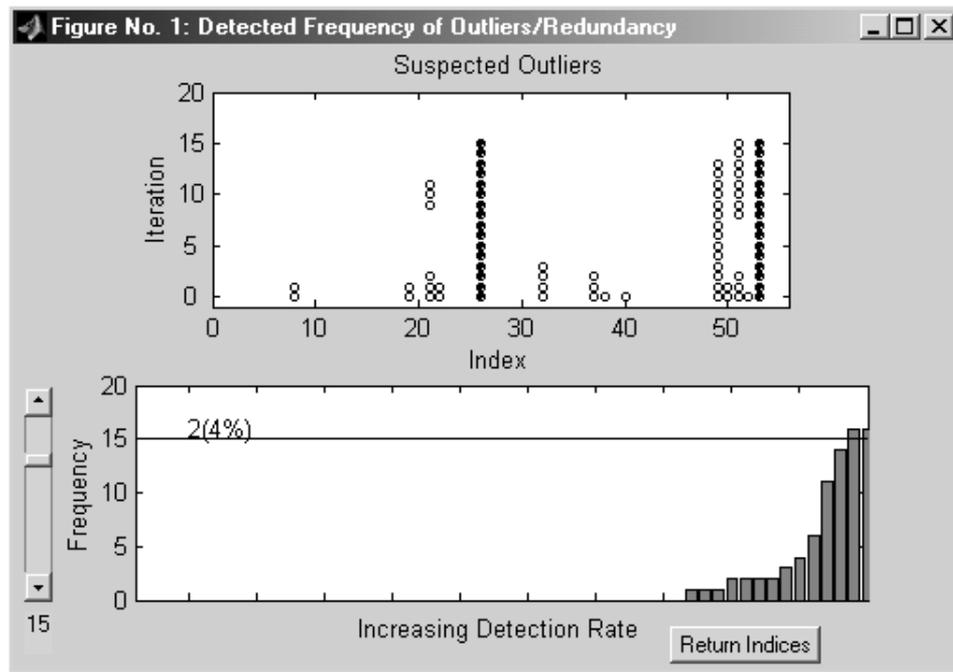


Figure 10.5: Frequency plots.

- *Use of variable noise level*

The incorporation of a variable noise level means that the value of ϵ varies over the input space. Note that in the user interface we prefer using parameter descriptions rather than the parameter name like ϵ . The variable ϵ information is obtained from a data file containing the value of ϵ corresponding to a learning data point. In order to avoid run time errors, the length of the variable ϵ vector is checked by the user interface. In Chapter 7 an example of this feature can be found.

10.4 Applications of Part III

The research design requirements discussed in Part III of this thesis were mainly aimed at investigating the capabilities the SVM may have in realising the objective. Therefore most of the implementations were intended for experimental purposes only. As the topics presented in Part III are still under investigation and only in an experimental stage, their applications have not been fully implemented in the software. There are two applications from these chapters operational in the current implementation. The transductive modelling approach and the error statistics for evaluating the overall performance of the model.

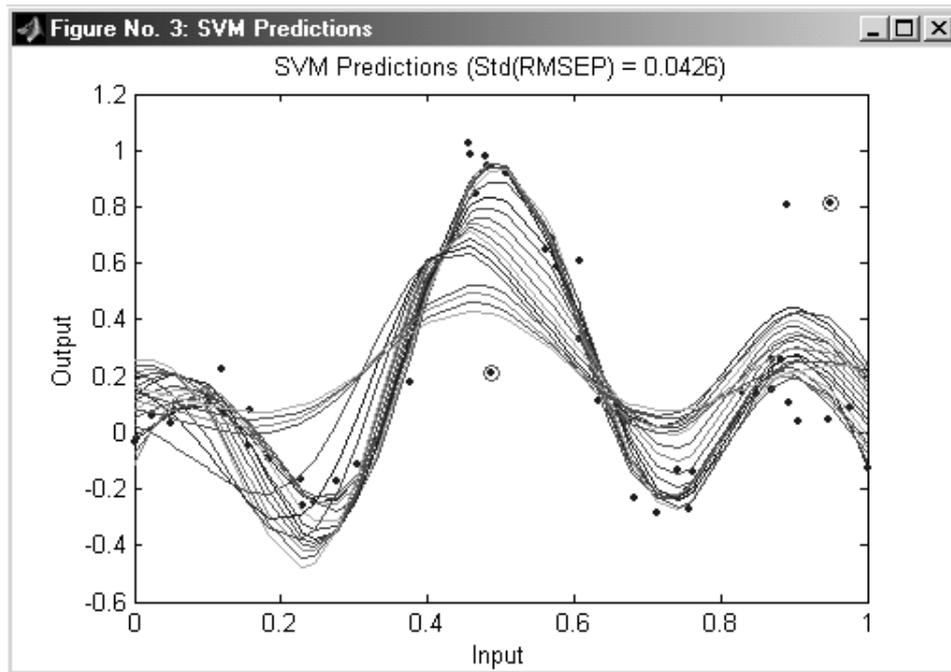


Figure 10.6: Predictions of SVMs and identified outliers.

10.4.1 Adaptivity

Of the three topics discussed in this chapter (novelty detection, adaptation levels, transduction), only transduction was implemented. For the implementation of the transductive learning we used Algorithm J. In our implementation the user is continuously given information about the current model as seen Figure 10.10, which shows the progress of the transduction at a point in time.

10.4.2 Self-diagnostic capabilities

In the chapter on self-diagnostic capabilities, we discussed a number of error measures for evaluation of the overall performance as well as investigated possible ways of assigning a measure of uncertainty to a prediction. It has been shown in Chapter 9 that the classical confidence limits implementation will give only information about the mean of the data. Therefore it would be better to use the error bar estimation. However, since the references to the error bar estimation were only recently published these results have not been fully investigated and no experiments have been done using the error bar. Therefore, only the overall error measures have been implemented at this time.

The overall error statistics discussed in Chapter 9 are automatically determined and are given as byproducts of the implemented SVM routines. The figure displaying

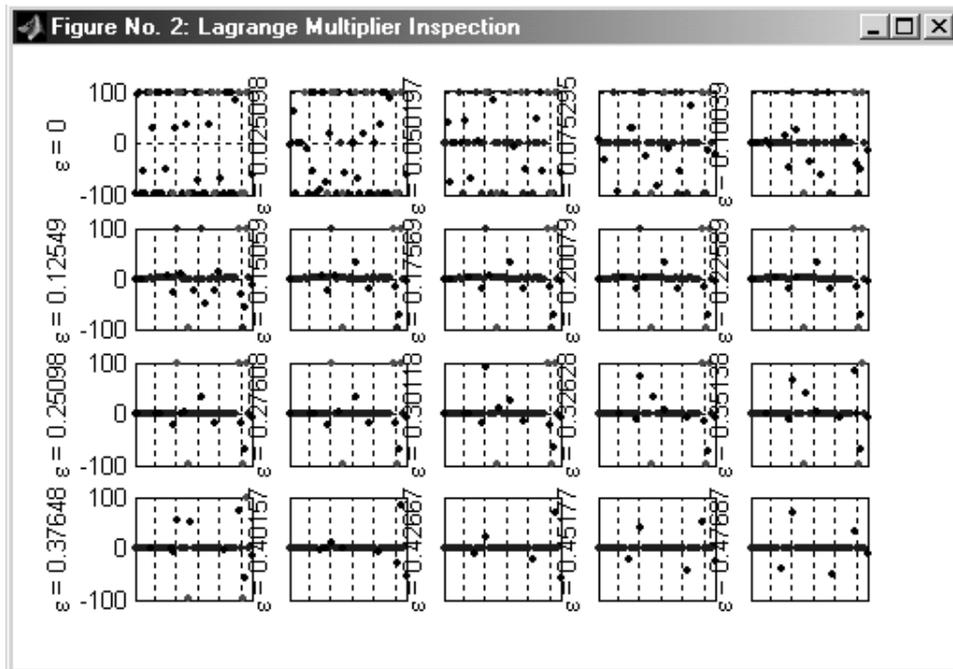


Figure 10.7: Lagrange Multiplier plots.

the results has an option to view the overall error measures as well as to investigate the residual distributions and display the normality plots.

Using the Performance Info menu bar in the figure displaying the results, the error statistics of the learning machine can be viewed. Figure 10.11 contains an example. Note that the number and percentage of support vectors are also displayed as well as the CPU time information.

The residual information is a figure that consists of three graphs, like in Figure 10.12: Firstly, a histogram showing the distribution of the residuals; Secondly the residuals plotted against the output data; Thirdly, the residuals plotted against one of the input variables. The figure also has a menu bar with which the user can choose which input variable is used in the third graph. In a separate figure the residual information for the test data is shown.

In Figure 10.13, the normal probability plot is shown. The plot for the test data is shown in a separate figure.

The interpretation of the information displayed by Figures 10.11, 10.12 and 10.13 has been discussed in Chapter 9.

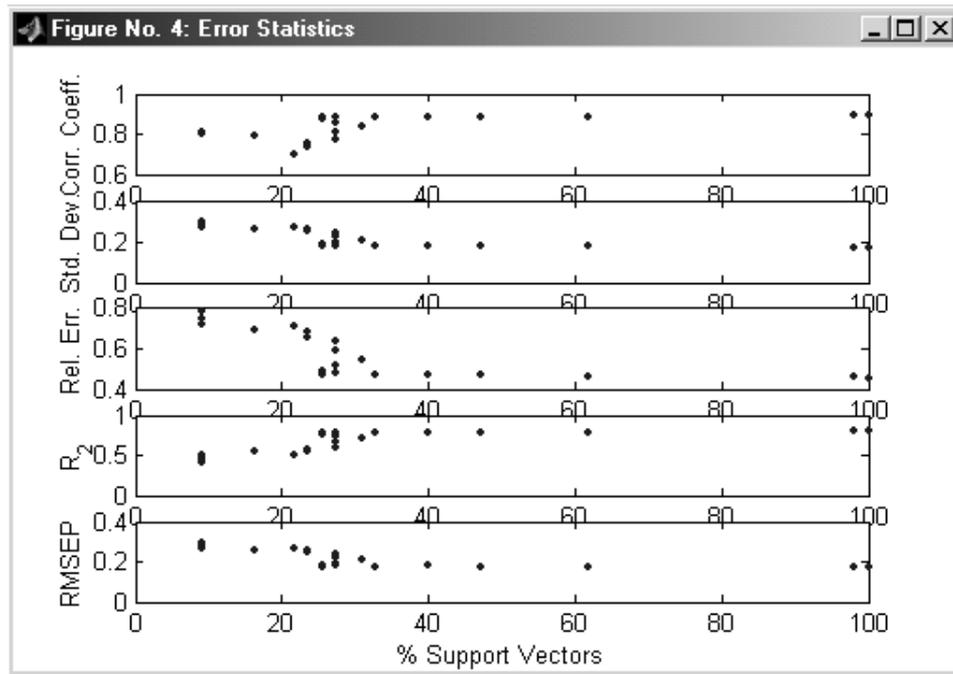


Figure 10.8: Error statistics for outlier detection iteration.

10.5 Miscellaneous features

In the implementation of the various elements that are discussed in Part I, II and III, a number of additional choices and applications was added. The additions include

- *Scaling method*

The scaling method can be set in the window for advanced setting (Figure 10.2). By default all data will be range scaled. The range scaling method was described in Chapter 3. Another scaling option is mean scaled. That means that the data will be scaled such that it has zero mean and one as standard deviation. Of course there is also the option to perform no scaling at all on the data.

- *Cached kernel for batch modelling*

In many application where multiple models are constructed, but the kernel and its parameters are fixed for all models, it is useful to cache the kernel. The kernel is determined only once, which saves computation time. The option to cache the kernel is available as an advanced setting in Figure 10.2.

- *Implementation of QP and LP optimisation*

It has been shown by several authors that the SVM method can also be written as a linear programming problem [10],[39],[78],[3]. By default the QP approach is used. However, if the user prefers the LP formulation, this advanced choice

The screenshot shows a dialog box titled "Input Sensitivity Analysis Results". It contains six input fields, each with a label and a numerical value:

- Weight for T cyl: 0.3
- Weight for T mold: 0.44
- Weight for t inj: 0.72
- Weight for sb: 1
- Weight for p hold: 0.5
- Weight for text: 1

At the bottom right of the dialog are two buttons: "OK" and "Cancel".

Figure 10.9: Entering sensitivity analysis information for a multi-dimensional scaling application.

is available as shown in Figure 10.2. More information about the LP problems can be found in [7].

- *Implementation of subset modelling methods*

It is important to notice that the computational speed of the SVM method scales with the number of data points in the learning data set [32]. For large data sets ($\ell > 1000$) the problem arises that the kernel matrix K becomes too large for the memory of standard computers. In [67] it was shown that for 50,000 examples, the kernel matrix would have 2.5×10^9 , that is 2.5 billion elements, which will require 20 GB of memory if an eight-byte floating point representation is used. Therefore, many implementations use decomposition approaches like the chunking method [35], [106] or sequential methods like the SMO method [69],[84]. We have implemented the chunking method with three possible ways of constructing the subset, namely disjoint subsets of size k , overlapping subsets of size k and $p\%$ of overlapping with the previous subset, and p randomly chosen subsets of size k . The parameters k and p are user defined parameters.

- *CPU time*

The software keeps track of the CPU time for modelling and all other operations. In the display results tool, this information can be viewed under the menu bar Performance Statistics.

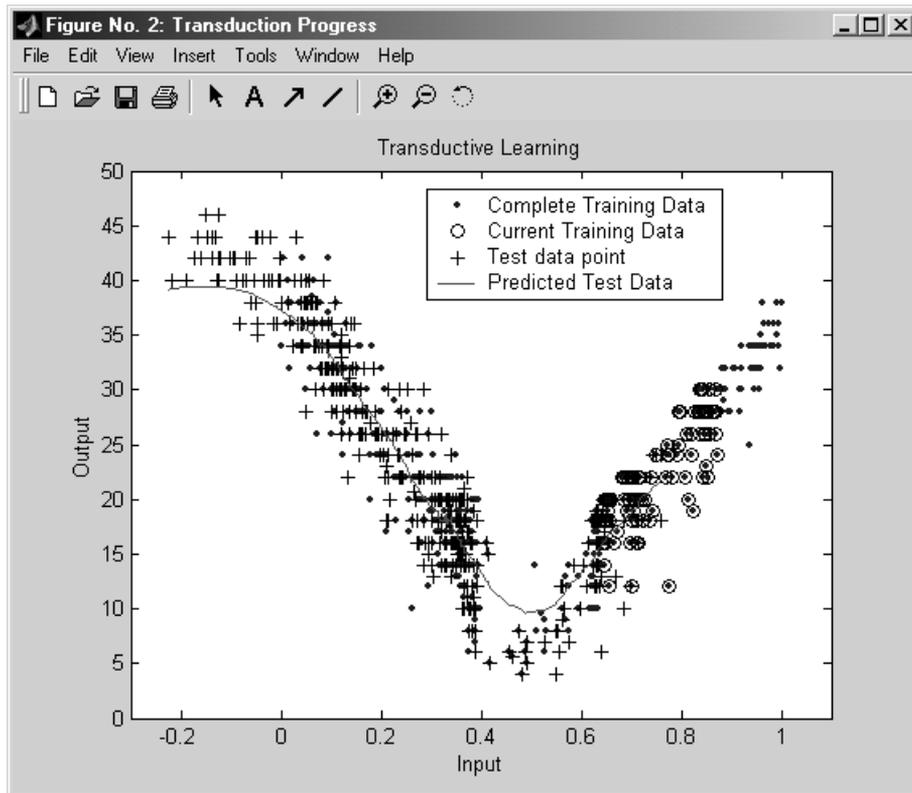


Figure 10.10: Snapshot of the transductive modelling phase.

- *Displaying and exporting results*

The software has an application to view and interpret the results obtained. The figure has two graphs. One displaying the learning data and the learning machine's predictions of it. The other shows the test data and their predicted values. In Figure 10.14 we show a typical figure window containing the results for an SVM learning machine. The figure has three additional menu bars, namely Display Features, Performance Information and Printing Options.

The Add Features menu bar enables the user to view the following features:

- Display tube or margin. For regression type problems, the ϵ -insensitive zone is added to the graph containing the learning data and their predicted values. In the case of classification problems, the margin is depicted by means of colouring the background of the different classes.
- Display Support Vectors. The support vectors are indicated in the graph containing the learning data.
- Reset Figure.

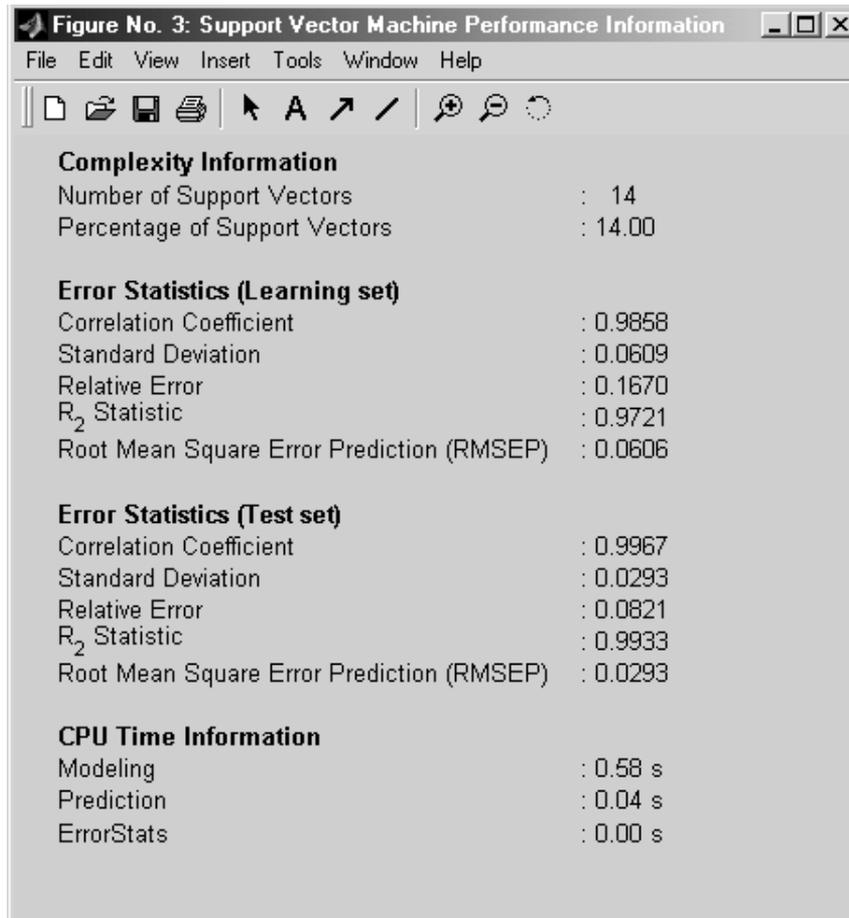


Figure 10.11: Display of the performance statistics of a model.

Sometimes adding some of the features listed above, may make interpretation of the graphs difficult. This option resets the graphs to their original state.

- Actual vs. Predicted. Often users prefer to view the actual output data plotted against the predicted.
- Parallel Coordinates. It is possible to view high-dimensional data sets using parallel coordinates [29]. Using this feature the user can visually observe trends in the behaviour of the problem.
- Lagrange Multipliers. The weights obtained from the Lagrange multipliers of the QP solution, are depicted in a separate window.

The various options available in the Performance Info menu bar have been discussed in detail in the previous section. We therefore give here an overview of

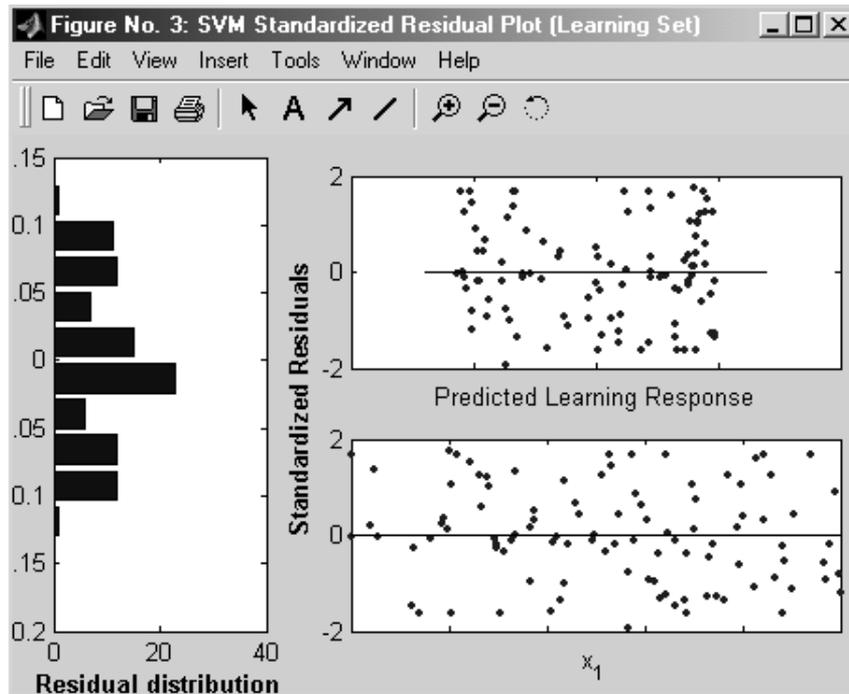


Figure 10.12: Display of residual information of a model.

all options

- SVM Parameters. The parameter settings used during the learning phase are displayed.
- Error Statistics. The support vector information, overall error measures and CPU time information are displayed.
- Error Distribution. The normal distribution plots for both the learning and test data are given.
- Residual Information. The residual plots for both the learning and test data are given.

Finally, the Print Option menu bar has the following options to save the figure

- Eps File.
- Jpeg File.
- Bitmap File.
- MATLAB Figure File.

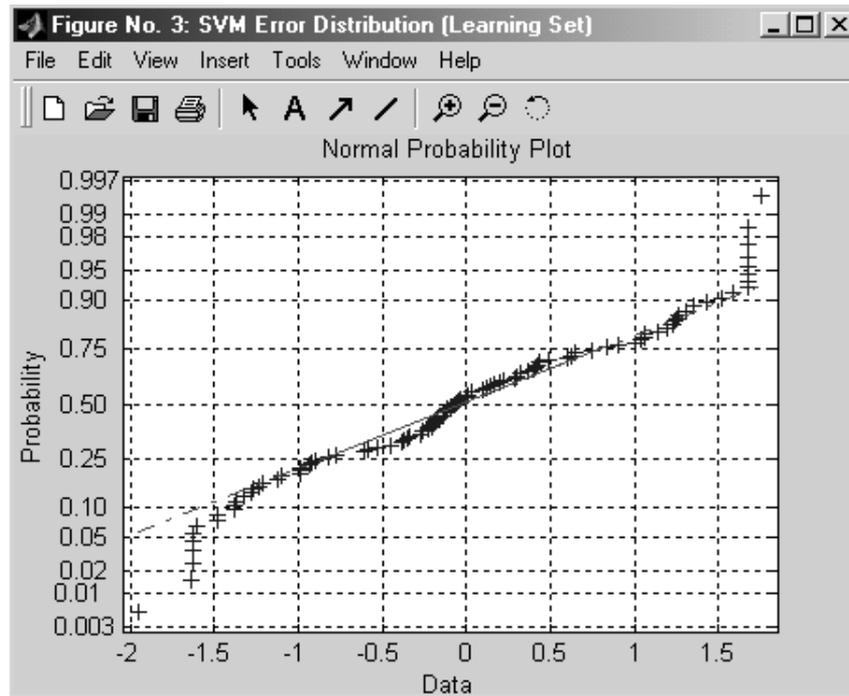


Figure 10.13: Display of the normality probability plots of a model.

10.6 Conclusion

In this chapter examples of the applications that have been implemented and examples of their use were given. The applications realising the learning requirements (Part I) and the application robustness requirements (Part II) have been fully implemented. However, only applications in the operational requirements (Part III) that are directly related to the learning process, namely overall error statistics and transductive learning, were implemented. The other applications involve implementations around the learning process that require more research.

One advantage of the user interface for the applications illustrated in this chapter, is that the user has a complete overview of which parameters to set and various options that are available to improve the learning process. Furthermore, the user not only has the option to optimise a number of the parameters but also has the ability to control and interpret the results obtained from the optimisation routines.

The results can be viewed in a window allowing the user to add and interpret various features like the ϵ -insensitive tube and support vectors as well as a number of performance information measures. An option to export results for reporting has also been included.

The implemented applications illustrated in this chapter form a basis from which an adaptive inferential sensor can be built. At The Dow Chemical Company the

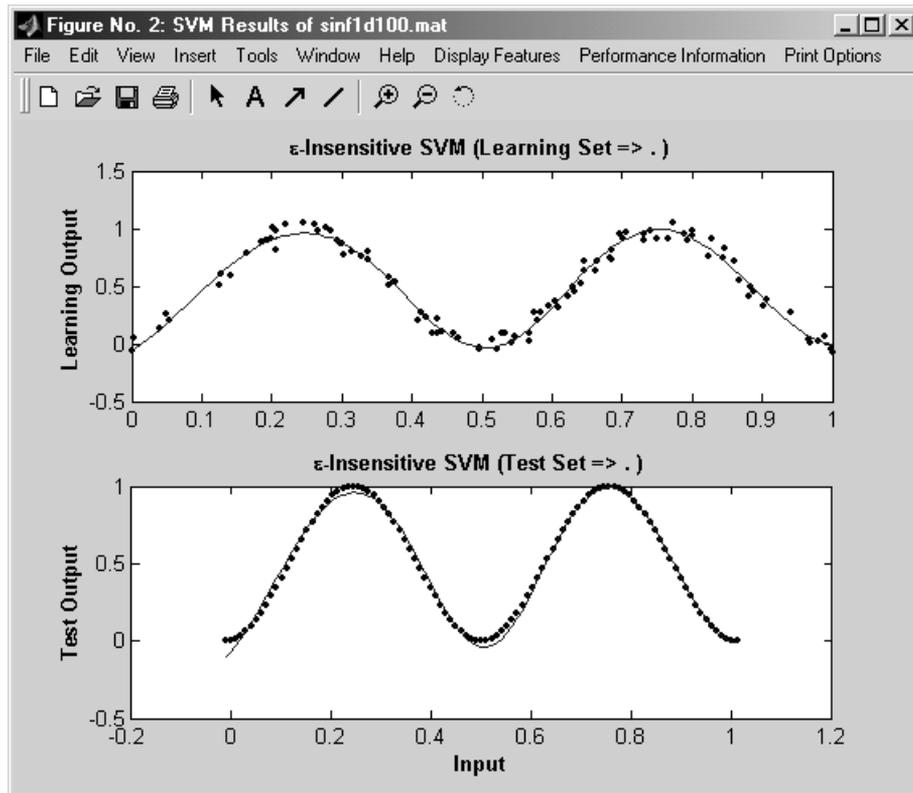


Figure 10.14: Display of the the results of an SVM learning machine.

applications are used in conjunction with other learning machines like NNs and GP [41] during inferential sensor development. In our opinion many features need to be added to the software. For example, a feature selection step using kernel PCA [79]. In the course of time, the software will also include other learning machines such that it will form a comprehensive modelling and data analysis tool.

Chapter 11

Conclusions

In this chapter we give a summary of the results, conclusions and implementations concerning each of the design requirements stated in Chapter 1. The design requirements are categorised into three main types which are presented in three parts of the design thesis.

In Part I on the learning requirements, we looked at those design requirements that define and influence the learning capabilities of the inferential sensor.

- *Complexity Control.* We have shown that implementing the SVM as a learning method enables the inferential sensor to control the complexity of the resulting model. The control can be either explicitly done using the ϵ -insensitive zone or implicitly using the ratio of support vectors (ν). We also gave algorithmic pseudocodes for the optimisation of the two parameters ϵ and ν . The SVM method is ideal for building inferential sensors that have direct control over the complexity of the models.
- *Working with high-dimensional data and spaces.* Due to the curse of dimensionality many learning methods experience loss of both computational and predictive performance. Using the implicit mapping to a higher dimensional space through kernel functions, the SVM partially overcomes the computational problem. Furthermore, since the SVM's learning capacity is not defined in terms of the number of dimensions but in terms of the VC-dimension, the curse of dimensionality with respect to the predictive performance is overcome to some extent. We further discussed two types of kernel functions and also gave algorithmic pseudocodes for optimisation of the kernel parameters. Therefore, the SVM method using the kernel trick enables the inferential sensor to work with high-dimensional data which is often encountered in the chemical industry.
- *Robustness.* The inferential sensor built on SVM-technology is made robust with respect to noise and outliers. This is done by using the ϵ -insensitive loss function. We further derived a heuristic for estimating the regularisation parameter C . This parameter is used as a trade-off between the error defined by the loss function and the complexity defined by the smoothness of the model.

In Part II, which is concerned with the application stability requirements, the design requirements which aim at improving the stability and performance of the inferential sensor are included.

- *Generalisation ability.* The interpolation and extrapolation abilities of the model are mainly defined by the kernel function used by the SVM method. We showed the advantages and disadvantages of using the two kernel function discussed in Part I. We further introduced the mixed kernel approach which improves the generalisation ability of the inferential sensor.
- *Data compression and outlier detection.* In recent years scientists expressed a need for model-based approaches to detect outliers and redundancy in data sets since classical approaches often fail due to the curse of dimensionality. It has also been suggested by many that the unique properties of SVMs could be used to develop such approaches. We investigated and developed on the basis of SVM-technology a model-based approach for outlier detection and data compression purposes. These new approaches enable the inferential sensor to keep up with the needs of the industry.
- *Incorporation of prior knowledge.* For the inferential sensor to be stable in unknown or low data density regions of the input space, it needs to incorporate prior knowledge. However, this can only be done if the model is able to generalise well. Since the mixed kernel approach has achieved just that, we can venture into the field of incorporating prior knowledge. We have shown in the design thesis how various kinds of prior knowledge can be incorporated into the SVM.

The Part III involves the operational requirements. These design requirements discussed here are aimed at making the inferential sensor trustworthy as well as increase its lifespan.

- *Adaptation.* In order to extend the lifespan of a inferential sensor it needs to become adaptive to changing conditions. That requires first of all the ability to detect novelty. We have shown how SVMs can be used to perform novelty detection and discussed various levels of adaptivity that can be implemented.
- *Self-diagnostic capabilities.* The inferential sensor needs to reflect its trustworthiness. We gave an overview of a number of overall performance statistics that can be used including a measure based on SLT, namely Vapnik's measure. One issue that still remains unanswered is the uncertainty level associated with a prediction. We investigated three possibilities and discussed their potential use.

Finally, in Part IV, the software implementations of the pseudocodes presented throughout the design thesis are shown. Furthermore, the effectiveness of the user interface and interactive application tools are illustrated.

These design requirements give us a framework from which an adaptive and intelligent inferential sensor can be developed. A schematic overview of this framework is given in Figure 11.1. The diagram has two main phases namely, off-line and online. The off-line phase has a learning part and a part consisting of theoretical knowledge

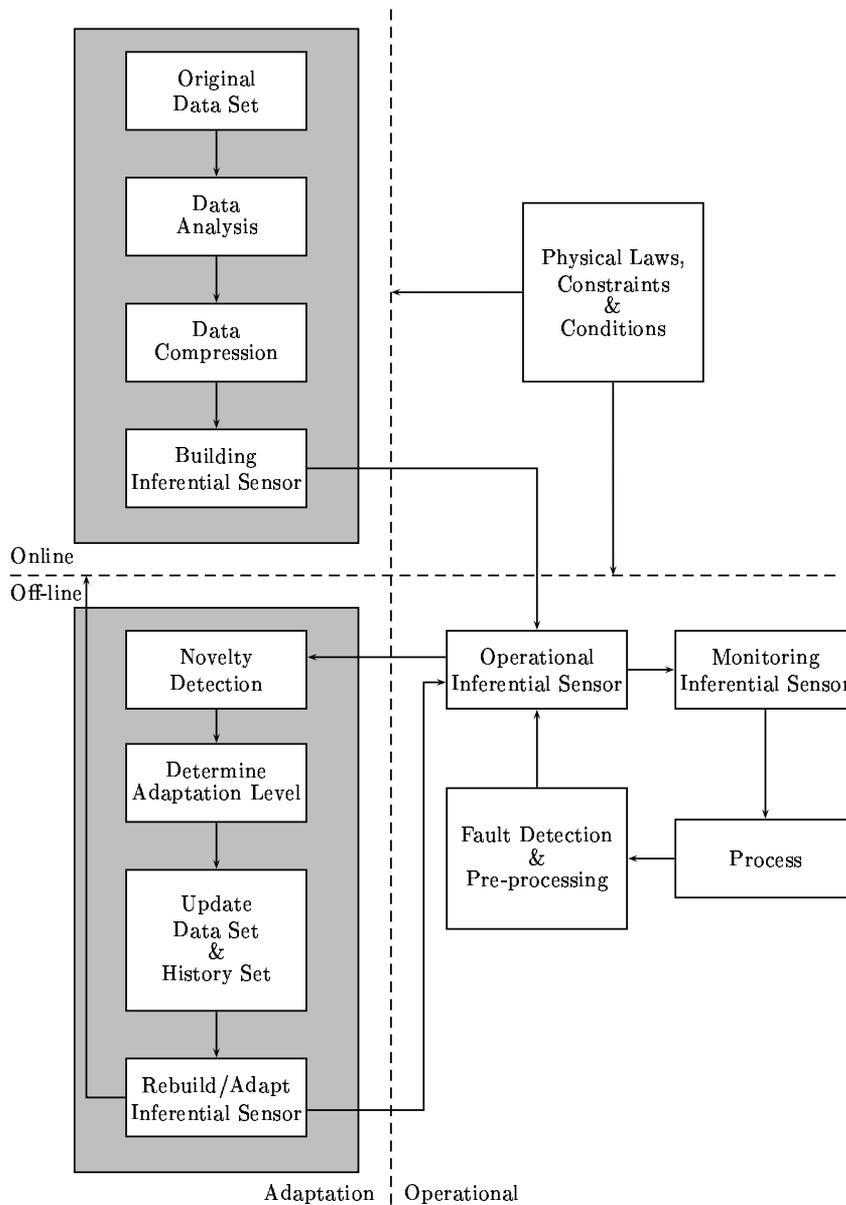


Figure 11.1: The Adaptive Online Inferential Sensor

and other information. The online phase consists of an operation part and a adaptation part. Each part has various steps and actions to be taken.

When building the inferential sensor in the off-line phase of Figure 11.1, a number of steps have to be performed: transforming the original data into a suitable data format, data analysis, data compression, and building a inferential sensor model from the learning data. The learning method uses a data set that has been analysed and reduced, and incorporates information from the real world like bounds on variables

and physical laws as well. The whole off-line phase is an iterative process in which the user often will return to the data analysis or even original data set to make some changes or adjustments. More than one model for the inferential sensor may also be built and used online. The methods currently used in the first two steps are based on classical statistics and are mostly linear methods. There are feature selection and data reduction tools that are nonlinear methods and work well for small samples. Nonlinear learning methods are becoming increasingly important every day. There are also indications that the SVM method can be used for feature selection [79].

The online operation inferential sensor predicts and evaluates the outcome of new data. After some time the process will give a measurement of the actual outcome of the new data. This measurement is tested for any faults and pre-processed. Using the predicted value and the measurement, the performance of the inferential sensor is monitored and results sent back to the process if necessary. All the physical information in the form of laws, constraints and conditions is known to the online operational model. Simultaneously, the inferential sensor sends all the necessary information through to the adaptive online mode. The adaptive mode of the inferential sensor tests online for any novel behaviour of the process. The level of adaptation is then determined. Several levels of adaptation could exist, varying from no adaptation to a complete off-line rebuilding of the inferential sensor.

There are various issues that require in our opinion a fair amount of further research. One person alone cannot address every aspect let alone solve all the problems that arise from it. In the conclusions of the various chapters we discussed them in short. However, there are a number broader issues that require some extra consideration.

- The whole concept of incorporating first principle information as a form of prior knowledge requires extensive research.
- In order to use SVMs for regression in the chemical industry the problem of calculating confidence limits of uncertainty levels needs to be solved. The statistical properties of SVMs also needs to be better understood.
- On the issue of adaptivity, we only investigated the potential of SVMs. A full and thorough investigation as well as development of the software still has to be done.
- There is one aspect of SVMs for regression that hampers its use in industry. The optimisation routines need to be adjusted to accommodate large-scale data sets. However most commercial and of-the-shelve packages do not allow the user access to the source code.
- In our opinion for inferential sensor development it is not a matter of using either SVMs, NNs or GPs. It is in fact a matter of using the method which is best suited for the particular problem in question. Therefore, we feel that there is a need to investigate combining and integrating these methods into a larger software package.

Bibliography

- [1] C.C. AGGARWAL, P.S. YU, *Outlier detection for high dimensional data*, Proceedings of the SIGMOD Conference, 2001.
- [2] ASPENTECH, *IQModel 1.0 User Guide*, 1998.
- [3] K.P. BENNET, *Combining support vector and mathematical programming methods for classification*, Advances in kernel methods, Support vector learning, B. Schölkopf, C.J. Burges and A.J. Smola (eds.), pp. 307-326, MIT Press, London, 1998.
- [4] B. BOSER, I. GUYON, V.N. VAPNIK, *A training algorithm for optimal margin classifiers*, Fifth Annual Workshop of Computational Learning Theory, pp. 144-152, Pittsburgh ACM, 1992.
- [5] V.L. BRAILOVSKY, O. BARZILAY, R. SHAHAVE, *On global, local, mixed and neighborhood kernels for support vector machines*, Pattern Recognition Letters 20, pp. 1183-1190, 1999.
- [6] B.D. BUNDAY, G.R. GARSIDE, *Optimisation methods in PASCAL*, Edward Arnold (Publishers) Ltd, London, 1987.
- [7] B.D. BUNDAY, G.R. GARSIDE, *Linear programming in PASCAL*, Edward Arnold (Publishers) Ltd, London, 1987.
- [8] C.J.C. BURGESS, B. SCHLKOPEF, *Improving the accuracy and speed of support vector learning machines*, Advances in Neural Information Processing Systems 9, pp. 375-381, Morgan Kaufmann, San Mateo, CA, 1997.
- [9] C. CAMPBELL, K.P. BENNET, *A linear programming approach for novelty detection*, Advances in Neural Information Processing Systems 13, pp. 395-401, MIT Press, 2001.
- [10] N. CRISTIANINI, J. SHAWE-TAYLOR, *An introduction to support vector machines, and other kernel-based learning methods*, Cambridge University Press, 2000.
- [11] V. CHERKASSKY, F. MULLIER, *Learning from data, Concepts, theory, and methods*, John Wiley, New York, 1998.

- [12] W. CHU, S.S. KEERTHI, C.J. ONG, *Bayesian inference in support vector regression*, Technical Report, CD-01-15, Control Division, Department of Mechanical Engineering, National University of Singapore, 2001.
- [13] E.F. CODD, S.B. CODD, T.S. CLYNCH, *Beyond decision support*, Computer World, July 26, 1993.
- [14] V. DHAR, R. STEIN, *Seven methods for transforming corporate data into business intelligence*, Prentice Hall, Upper Saddle River, NJ, 1997.
- [15] H.W. ENGL, M. HANKE, A. NEUBAUER, *Regularisation of inverse problems*, Kluwer Academic Publishers, Hingham, MA, 1996.
- [16] T. EVGINIOU, M. PONTIL, T. POGGIO, *A unified framework for regularization networks and support vector machines*, A.I. Memo 1645, Artificial Intelligence Laboratory, MIT, MA, 1999.
- [17] R. FLETCHER, *Practical methods of optimization, Second edition*, John Wiley & Sons, New York, 1989.
- [18] S. FORTUNE, *Voronoi diagrams and Delaunay triangulations*, Computing in Euclidean Geometry, D.Z. Du and F. Hwang (eds.), pp. 193-233, World Scientific, 1992.
- [19] A. GAMMERMAN, V. VOVK, V.N. VAPNIK, *Learning by transduction*, Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, pp. 148-155, Morgan Kaufmann, San Francisco, CA, 1998.
- [20] J.B. GAO, S.R. GUNN, C.J. HARRIS, M. BROWN, *A Probabilistic Framework for Support Vector Machine Regression and Error Bar Estimation*, Machine Learning 46, pp. 71-89, 2002.
- [21] G.H. GOLUB, U. VON MATT, *Tikhonov regularization for larger scale problems*, Scientific Computing, G.H. Golub, S.H. Lui, F.T. Luk and R.J. Plemmons (eds.), Springer, Berlin, 1997.
- [22] A.V. GRIBOK, J.W. HINES, R.E. UHRIG, *Use of kernel based techniques for sensor validation in nuclear power plants*, Proceeding of the International Topical Meeting on Nuclear Plant Instrumentation, Controls, and Human-Machine Interface Technologies, Washington, DC, November, 2000.
- [23] P.C. HANSEN, *The L-curve and its use in the numerical treatment of inverse problems*, Computational inverse problems in electrocardiology, P. Johnston (ed.), pp. 119-142, WIT Press, Southampton, 2001.
- [24] T.J. HASTIE, R.J. TIBSHIRANI, *Generalized linear models*, Chapman and Hall, London, UK, 1990.
- [25] P. HAYTON, B. SCHÖLKOPF, L. TARASSENKO, P. ANUZIS, *Support vector novelty detection applied to jet engine vibration spectra*, Advances in Neural Information Processing Systems 12, pp. 946-952, MIT Press, 2000.

- [26] J.W. HINES, A.V. GRIBOK, I. ATTIEH, R.E. UHRIG, *The use of regularization in inferential measurements*, Enlarged Halden programme Group Meeting, Loen, Norway, May, 1999.
- [27] P.J. HUBER, *Robust statistics*, John Wiley & Sons, New York, 1981.
- [28] P.J. HUBER, *Robust statistics: a review*, Annals of Statistics 43, p. 1041, 1972.
- [29] A. INSELBERG, T. CHOMUT, *Convexity algorithms in parallel coordinates*, Journal of the Association for Computing Machinery, 43(4), pp. 765-801, 1987.
- [30] T.S. JAAKKOLA, D. HAUSSLER, *Probabilistic kernel regression models*, Proceedings of the 1999 Conference on Artificial Intelligence and Statistics, 1999.
- [31] B. JEPSON, A. COLLINS, A. EVANS, *Post-neural network procedure to determine expected prediction values and their confidence limits*, Neural Computation and Applications 1, pp. 224-228, 1993.
- [32] T. JOACHIMS, *Making large-scale support vector machine learning practical*, Advances in kernel methods, Support vector learning., B. Schölkopf, C.J. Burges and A.J. Smola (eds.), pp.169-184, MIT Press, London, 1998.
- [33] E.M. JORDAAN, *Development of adaptive online soft sensors*, master's thesis TU Eindhoven, ISBN 90-5282-993-4, 1999.
- [34] E.M. JORDAAN, G.F. SMITS, *Estimation of regularization parameter for support vector regression*, Proceedings of the 2002 IEEE World Conference on Computational Intelligence, pp.2785-2791, 2002.
- [35] L. KAUFMAN, *Solving the quadratic programming problem arising in support vector classification*, Advances in kernel methods, Support vector learning., B. Schölkopf, C.J. Burges and A.J. Smola (eds.), pp. 147-168, MIT Press, London, 1998.
- [36] M. KEARNS, *The computational complexity of machine learning*, MIT Press, London, 1990.
- [37] M. KEARNS, L. VALIANT, *Cryptographic limitations on learning, formulae and finite automata*, Journal of the ACM, 41(1), pp. 67-95, 1994.
- [38] M.J. KEARNS, U.V. VAZIRANI, *An introduction to computational learning theory*, MIT Press, London, 1994.
- [39] V. KECMAN, *Learning and soft computing, Support vector machines, neural networks, and fuzzy logic models*, MIT Press, London, 2001.
- [40] J.D. KEELER, R.B. FERGUSON, *Commercial Applications of SoftSensors[®]: The Virtual On – line Analyzer[®] and the Software CEM[®]*, Proceedings of the International Forum for Process Analytical Chemistry, Orlando, FL, January, 1996.

- [41] A. KORDON, G.F. SMITS, E.M. JORDAAN, E. RIGHTOR, *Robust inferential sensors based on integration of genetic programming, analytical neural networks, and support vector machines*, Proceedings of the 2002 World Conference on Computational Intelligence, pp. 896-901, 2002.
- [42] E. KREYSZIG, *Introductory functional analysis with applications*, John Wiley & Sons, New York, 1978.
- [43] E. KREYSZIG, *Advanced engineering mathematics, Seventh edition*, John Wiley & Sons, New York, 1993.
- [44] J.T-Y. KWOK, *The evidence framework applied to support vector machines*, IEEE Transaction of Neural Networks, 20(10), 2000.
- [45] J.T-Y. KWOK, *Linear dependency between epsilon and the input noise in epsilon-support vector regression*, Proceedings of the International Conference on Artificial Neural Networks (ICANN'01), pp. 405-410, Vienna, Austria, August, 2001.
- [46] M.H. LAW, J.T. KWOK, *Applying the bayesian evidence framework to ν -support vector regression*, Proceedings of the European Conference on Machine Learning, Brugge, September, 2001.
- [47] P. LERAY, P. GALLINARI, *Feature selection with neural networks*, Technical report LIP6 1998/012, LIP6, 1998.
- [48] C. LI, W.H. WONG, *Model-based analysis of oligonucleotide arrays: expression index computation and outlier detection*, Proc Natl Acad Sci USA 98, pp. 31-36, 2001.
- [49] W.L. LUYBEN, *Process modelling, simulation and control for chemical engineers*, McGraw-Hill Education, New York, 1999.
- [50] D.J. MACKAY, *Gaussian processes, A replacement for neural networks*, Neural Information Processing Systems Tutorial, Cambridge University, 1997.
- [51] O.L. MANGASARIAN, *Nonlinear programming*, SIAM Classic in Applied Mathematics 10, Philadelphia, 1994.
- [52] B.F.J. MANLY, *Multivariate statistical methods, A primer, Second edition*, Chapman & Hall, London, 1998.
- [53] G. MARTIN, *Neural network applications for prediction, control and optimization*, Advances in Instrumentation and Control 50, ISA, 1995.
- [54] *MATLAB[®] Reference Guide*, The MathWorks, Inc., Natick (MA), 1993.
- [55] D. MATTERA, S. HAYKIN, *Support vector machines for dynamic reconstruction of a chaotic system*, Advances in kernel methods, Support vector learning., B. Schölkopf, C.J. Burges and A.J. Smola (eds.), pp. 211-242, MIT Press, London, 1998.

- [56] D.C. MONTGOMERY, E.A. PECK, *Introduction to linear regression analysis, Second Edition*, John Wiley & Sons, New York, 1992.
- [57] S. MORRISON, *The importance of automatic design technique in implementating neural net intelligent sensors*, ISA'96, 1996.
- [58] K.-R. MÜLLER, A.J. SMOLA, G. RÄTSCH, B. SCHÖLKOPF, J. KOHLMORGEN, V.N. VAPNIK, *Predicting time series with support vector machines*, Proceedings of the International Conference on Artificial Neural Networks, pp. 999-1004, Springer Verlag, Berlin, 1997.
- [59] K.-R. MÜLLER, A.J. SMOLA, G. RÄTSCH, B. SCHÖLKOPF, J. KOHLMORGEN, V.N. VAPNIK, *Using support vector machines for time series prediction*, Advances in kernel methods, Support vector learning., B. Schölkopf, C.J. Burges and A.J. Smola (eds.), pp. 243-254, MIT Press, London, 1998.
- [60] K.-R. MÜLLER, S. MIKA, G. RÄTSCH, K. TSUDA, B. SCHÖLLKOPF, *An introduction to kernel-based learning algorithms*, IEEE Transaction on Neural Networks, 2(2), pp. 181-201, 2001.
- [61] N. MURATA, K.-R. MÜLLER, A. ZIEHE, S. AMARI, *Adaptive on-line learning in changing environments*, Advances in Neural Processing Information Systems 9, p. 599, MIT Press, 1997.
- [62] N. MURATA, M. KAWANABE, A. ZIEHE, K.-R. MÜLLER, S. AMARI, *Online learning in changing environments with applications in supervised and unsupervised learning*, Neural Networks 15, pp. 743-760, 2002.
- [63] R.M. NEAL, *Regression and classification using gaussian process priors (with discussion)*, Bayesian statistics, J.M. Bernardo, et al. (eds.), pp. 475-201, Oxford University Press, London, 1998.
- [64] R. NEEKLAKANTAN, J. GUIVER, *Applying neural networks*, Hydrocarbon processing 9, 1998.
- [65] P. NIYOGI, F. GIROSI, T. POGGIO, *Incorporating prior information in machine learning by creating virtual examples*, Proceedings of the IEEE, 86(11), pp. 2196-2209, 1998.
- [66] A. O'HAGAN, *Bayesian inference*, Bayesian Inference 2B, 1994.
- [67] E. OSUNA, R. FREUND, F. GIROSI, *Support vector machines: Training and applications*, A.I. Memo No. 1602, MIT, Cambridge, MA, 1997.
- [68] R.J. PELL, *Multiple outlier detection for multivariate calibration using robust statistical techniques*, Chemometrics and Intelligent Laboratory Systems 52, pp. 87-104, 2000.
- [69] J.C. PLATT, *Fast training of Support Vector Machines using Sequential Minimal Optimization*, Advances in kernel methods, Support vector learning., B. Schölkopf, C.J. Burges and A.J. Smola (eds.), pp. 185-208, MIT Press, London, 1998.

- [70] S.J. QIN, H. YUE, R. DUNIA, *Self-validating inferential sensors with application to air emission monitoring*, Ind. Eng. Chem. Res. 36, pp. 1675-1685, 1997.
- [71] J.A. RICE, *Mathematical statistics and data analysis, Second edition*, Duxbury Press, Belmont, CA, 1995.
- [72] J. RISSANEN, *Modelling by shortest data description*, Automatica 14, pp. 465-471, 1978.
- [73] B. SCHÖLKOPF, P. BARTLETT, A.J. SMOLA, R. WILLIAMSON, *Support vector regression with automatic accuracy control*, Proceedings of the 8-th International Conference on Artificial Neural Networks, Perspective in Neural Computation, Springer-Verlag, Berlin, 1998.
- [74] B. SCHÖLKOPF, P. BARTLETT, A.J. SMOLA, R. WILLIAMSON, *Shrinking the tube: a new support vector regression algorithm*, Advances in Neural Information Processing Systems 11, pp. 330-336, MIT Press, Cambridge, 1999.
- [75] B. SCHÖLKOPF, C.J. BURGESS, A.J. SMOLA, *Advances in Kernel methods, Support Vector learning*, MIT Press, London, 1998.
- [76] B. SCHÖLKOPF, S. MIKA, C.J.C. BURGESS, P. KNIRSCH, K.-R. MÜLLER, G. RÄRSCH, A.J. SMOLA, *Input space vs. feature space in kernel-based methods*, IEEE Transactions on Neural Networks, 1999.
- [77] B. SCHÖLKOPF, P. SIMARD, A. SMOLA, V.N. VAPNIK, *Prior knowledge in support vector kernels*, Advances in Neural Processing Systems 10, MIT Press, Cambridge, MA, 1998.
- [78] B. SCHÖLKOPF, A.J. SMOLA *Learning with kernels, Support vector machines, regularisation, optimisation and beyond*, MIT Press, London, 2002.
- [79] B. SCHÖLKOPF, A.J. SMOLA, K.-R. MÜLLER, *Kernel principal component analysis*, Advances in kernel methods, Support vector learning., B. Schölkopf, C.J. Burgess and A.J. Smola (eds.), pp. 327-352, MIT Press, London, 1998.
- [80] B. SCHÖLKOPF, R. WILLIAMS, A.J. SMOLA, J. SHAW-TAYLOR, J. PLATT, *Support vector method for novelty detection*, Neural Information Processing Systems 12, pp. 582-588, MIT Press, 2000.
- [81] X. SHAO, *Model selection using statistical learning theory*, doctoral thesis proposal, unpublished, 1997.
- [82] R. SHAO, E.B. MARTIN, J. ZHANG, A.J. MORRIS, *Confidence bounds for neural network representations*, Computers Chemical Engineering, 21 Suppl., pp. S1173-S1178, 1997.
- [83] A.J.C. SHARKEY (Ed.), *Combining artificial neural nets, ensemble and modular multi-net systems*, Springer, London, 1999.

- [84] S.K. SHEVADE, S.S. KEERTHI, C. BHATTACHARYYA, K.R.K. MURTHY, *Improvements to SMO algorithm for SVM reegression*, IEEE Transactions on Neural Networks, 11(5), pp. 1188-1193, 2000.
- [85] G.F. SMITS, personal communication, 2002.
- [86] G.F. SMITS, E.M. JORDAAN, *Using mixtures of polynomial and RBF kernels for support vector regression*, Proceedings of the 2002 IEEE World Conference on Computational Intelligence, pp. 2192-2198, 2002.
- [87] A.J. SMOLA, *Regression estimation with support vector learning machines*, master's thesis TU Berlin, 1996.
- [88] A.J. SMOLA, *Learning with kernels*, Doctoral thesis TU Berlin, 1998.
- [89] A.J. SMOLA, N. MURATA, B. SCHÖLKOPF, K.-R. MÜLLER, *Asymptotically optimal choice of ϵ -loss for support vector machines*, Proceedings of the 8th International Conference on Artificial Neural Networks, Perspectives in Neural Computing, pp. 105-110, Springer-Verlag, Berlin, 1998.
- [90] A.J. SMOLA, B. SCHLKOPF, *On a kernel-based method for pattern recognition, regression, approximation and operator inversion*, Algorithmica 22, pp. 211-231, 1998.
- [91] A.J. SMOLA, B. SCHLKOPF, *A tutorial on support vector regression*, NC2-TR-1998-030, NeuroColt2 Technical Report Series, 1998.
- [92] A.J. SMOLA, B. SCHLKOPF, *From regularization operators to support vector kernels*, Advances in Neural Information Processing Systems 10, pp. 343-349, San Mateo, CA, 1998.
- [93] A.J. SMOLA, B. SCHLKOPF, K.-R. MÜLLER, *The connection between regularization operators and support vector kernels*, Neural Networks 11, pp. 637-649, 1998.
- [94] P. SOLLIC, *Probabilistic interpretations and Bayesian methods for support vector machines*, Proceedings of the International Conference on Artificial Neural Networks (ICANN'99), pp. 91-96. IEE Publications, 1999.
- [95] R.H. SPRAGUE, JR., E.D. CARLSON, *Building effective decision support systems*, Prentice-Hall, Inc., London, 1982.
- [96] STAN ACKERMANS INSTITUUT, *Notitie: Op weg naar promotie op proefontwerp*, TU Eindhoven, Eindhoven, 1994.
- [97] G. STRANG, *Linear algebra and its applications, Third edition*, Harcourt Brace Jovanovich College Publications, New York, 1988.
- [98] J. TANG, Z. CHEN, A. FU , D. CHEUNG, *A robust outlier detection scheme in large data sets*, Proceedings of the Sixth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'02), Taipei, 6-8 May, 2002.

- [99] D.M.J. TAX, A. YPMA, R.P.W. DUIN, *Support vector data description applied to machine vibration analysis*, Proceedings of the Fifth Annual Conference on the Advanced School for Computing and Imaging, Heijen, The Netherlands, pp. 398-405, 1999.
- [100] M.T. THAM, G.A. MONTAGUE, J.J. MORRIS, P.A. LANT, *Soft-sensors for process estimation and inferential control*, Journal of Process Control 11, January, 1991.
- [101] A.N. TIKHONOV, *On solving ill-posed problems and method of regularization*, Doklady Akademii Nauk USSR 153, pp. 501-504, 1963.
- [102] N. TISHBY, E. LEVIN, S.A. SOLLA, *Consistent inference of probabilities in layered Networks: Predictions and generalization*, Proceedings of International Joint Conference of Neural Networks, 2, pp. 403-409, IEEE, Washington, DC, 1989.
- [103] K. TSUDA, G. RÄTSCHE, S. MIKA, K.-R. MÜLLER, *Learning to predict the leave-one-out error of kernel-based classifiers*, Proceedings of the International Conference on Artificial Neural Networks (ICANN'01), pp. 331-338, Springer Lecture Notes in Computer Science, 2001.
- [104] L.G. VALIANT, *A theory of the learnable*, Communications of the ACM, 27(11), pp. 1134-1142, 1984.
- [105] V.N. VAPNIK, L.-Y. BOTTOU, *Local algorithms for pattern recognition and dependencies estimation*, Neural Computation, 5(6), pp. 893-908, 1993.
- [106] V. VAPNIK, *The nature of statistical learning theory*, Springer, New York, 1995.
- [107] V.N. VAPNIK, S.E. GOLLOWICH, A. SMOLA, *Support vector method for function approximation, regression estimation and signal processing*, Advances in Neural Information Processing Systems 9, pp. 281-287, 1997.
- [108] V.N. VAPNIK, *Statistical learning theory*, John Wiley & Sons, 1998.
- [109] V.N. VAPNIK, E. LEVIN, Y. LECUN, *Measuring the VC dimension of a learning machine*, Neural Computation, 10(5), 1994.
- [110] C.K. WILLIAMS, *Prediction with Gaussian processes: From linear regression to linear prediction and beyond*, Learning in Graphical Models, M.I. Jordan (ed.), pp. 295-621, MIT Press, Cambridge, MA, 1998.
- [111] C.K.I. WILLIAMS, M. SEEGER, *The effect of the input density distribution on kernel-based classifiers*, Proceedings of the Seventeenth International Conference on Machine Learning, Morgan Kaufmann, Philadelphia, 2000.
- [112] X. XU, J.W. HINES, R.E. UHRIG, *Sensor validation and fault detection using neural networks*, Proceedings of the Maintenance and Reliability Conference (MARCON 99), Gatlinburg, TN, May 10-12, 1999.

- [113] L.A. ZADEH, *Fuzzy sets*, Information and Control 8, pp. 338-353, 1965.
- [114] L.A. ZADEH, *Soft computing and fuzzy logic*, IEEE Software, November, pp. 48-56, 1994.

Samenvatting

Het hoofddoel van het onderzoek was om de technologie die nodig is voor de bouw en het gebruik van adaptieve *inferential sensors* te onderzoeken en verder te ontwikkelen. Inferential sensors zijn wiskundige modellen die in de industrie gebruikt worden om de uitkomst van processen te voorspellen. Aangezien de processen niet statisch zijn, is het nodig dat de inferential sensors adaptief zijn ten opzichte van veranderende omstandigheden teneinde een langere periode toepasbaar te blijven. De technologie die nodig is voor de ontwikkeling van een adaptieve inferential sensor vereist de integratie van de operationele- en applicatie-eisen met de software die gebruikt wordt voor modellering. Daarom werd als een eerste stap in dit ontwerpproject de volgende ontwerpeisen geïdentificeerd: (1) controle over de complexiteit, (2) vermogen om hoog-dimensionale data te gebruiken, (3) robuustheid, (4) generalisatievermogen, (5) data-reductie en *outlier*-detectie (6) inachtneming van a priori kennis (voorkennis), (7) zelfdiagnose en (8) adaptief vermogen.

Een nieuw soort leermachine, de *Support Vector Machine* (SVM), wordt gebruikt omdat deze het beste aan de eerste drie ontwerpeisen voldoet. Deze drie ontwerpeisen karakteriseren het leervermogen waaraan inferential sensors moeten voldoen. Dit wordt besproken in Deel I van het proefontwerp. In het proefontwerp hebben we ons geconcentreerd op regressie-problemen, omdat regressie de meest algemene toepassing is die in de chemische industrie voorkomt.

De ontwerpeisen (4), (5) en (6) worden geassocieerd met de stabiliteitseisen in applicaties die in Deel II van het proefontwerp worden besproken. In dit deel wordt een nieuwe soort kernfunctie geïntroduceerd. Het is een combinatie van een radiale basisfunctie en een polynoomkern die het generalisatievermogen van het inferential sensor model, gebouwd met behulp van SVM technologie, verhoogt. Het verbeterde generalisatievermogen stelt ons nu in staat om bepaalde soorten voorkennis, die algemeen verkrijgbaar zijn in de chemische industrie, te gebruiken. In onze implementatie van de data-reductie en outlier-detectie applicaties maken we gebruik van een nieuwe model-gebaseerde aanpak om outliers en overbodige data te herkennen.

In Deel III van het proefontwerp dat handelt over de operationele eisen, worden de laatste twee ontwerpeisen, (7) en (8), besproken. Wij hebben verschillende manieren onderzocht om de functionering van een inferential sensor te evalueren. Voor een inferential sensor is het niet alleen van belang om zijn eigen functionering te controleren en een diagnose te stellen, maar ook dat er een niveau van onzekerheid wordt toegekend aan een voorspelling. In het proefontwerp geven we een overzicht van een aantal foutstatistieken om de globale functionering te meten. Daarnaast bespreken wij ook

een drietal mogelijkheden voor de berekening van een onzekerheidsniveau behorend bij een gegeven voorspelling. We verstrekken een aantal adaptatieniveaus en geven daarnaast een methode voor *novelty*-detectie, gebaseerd op SVM technologie.

Voor de ontwikkeling van adaptieve inferential sensors is het nodig dat de parameters die de leermachine gebruikt, gemakkelijk bepaald of geschat kunnen worden. In Deel I van het proefontwerp stellen we een heuristische methode voor om de regulariseringsparameter van de SVM voor regression te schatten. Verder worden algoritmische pseudocoden voor de optimalisering van de parameters gegeven in de hoofdstukken waar deze parameters worden besproken.

Tenslotte, in Deel IV van het proefontwerp wordt een software tool gepresenteerd die de SVM als leeralgoritme combineert met de ontwerpeisen. Een aantal ontwerpeisen, zoals outlier-detectie en data-reductie, zijn geïmplementeerd als aparte applicatie tools. Uiteraard zijn de algoritmen voor de optimalisering van de SVM parameters ook geïmplementeerd. Verder heeft de software tool een *user interface* die de gebruiker assisteert om parameters in te stellen of te optimaliseren evenals andere modellerings keuzes te maken. De gebruiker wordt daarnaast in staat gesteld om de resultaten te onderzoeken, te analyseren en te interpreteren. De software is al een geruime tijd in gebruik bij The Dow Chemical Company en voorbeelden van de toepassingen worden gegeven.

Acknowledgements

Since I can remember I wanted to study abroad and obtain a Ph.D. I also had a keen interest in the usefulness of mathematics and chemistry from an early age. My wildest dreams came true when I got the opportunity to study in Holland and conduct my research in mathematics within the chemical industry. Here I want to give my gratitude to those who made it possible for me to come this far.

Dr. Guido Smits deserves a special word of thanks. Not only did he supervise the research project, he also taught me to appreciate the elegance of theory. I look back with joy to our lengthy discussions on Support Vector Machines (SVMs) and the insight I obtained from his broad knowledge on machine-learning issues. I sincerely hope that in the coming years we will work together on many occasions where I can further learn much from his knowledge. Our numerous debates on world politics and social-economical issues have been equally enjoyed.

I also want to express my gratitude to Prof.dr. Klaus-Robert Müller, Prof.dr. Emile Aarts, Prof.dr. Jaap Wessels, and the rest of the review-board for their interest, useful remarks and willingness to be part of this research. Dr. Arthur Kordon, Dr. Jaap den Doelder, and Dr. Randy Pell of The Dow Chemical Company, thank you for your keen interest in my work, your support, and valuable comments. Arthur, thanks for your suggestions on how to make the software more efficient and for finding those elusive bugs.

To all my colleagues at Dow-PTC 2 in Terneuzen, a special thank you for your hospitality and support as well as your patience in correcting my Dutch. I specially want to thank you for letting me speak Afrikaans, my mother tongue, on Fridays. I hope that we can continue this habit in the future.

I want to thank my family and friends, the Dutch and South-Africans, from the bottom of my hart. Without you I would not have been able to come to Europe or have been able to stick to my ambitions. A special word of thanks goes to my parents, Bert and Martie Jordaan, for everything they have done for me. You not only stimulated me to use the wit and talents bestowed upon me, but also supported me wholeheartedly in every decision I have ever made.

I know that the knowledge and sense I have is only temporary. At any moment my Maker could take it away from me. Therefore I endeavour to be a worthwhile human being.

Dankwoord

Sedert ek kan onthou wou ek in die buiteland gaan studeer het en 'n doktersgraad behaal het. Van jongs af het die toepaslikheid van wiskunde en chemie my gefassineer. My stoutste verwagtinge is egter oortref toe ek die geleentheid gekry het om in Nederland my navorsing in wiskunde te doen, en dit boonop binne die chemiese industrie. Hiermee wil ek die mense bedank wat daartoe bygedra het dat ek 'n sukses daarvan kon maak.

Dr. Guido Smits verdien hierby 'n besonder woord van dank. Nie alleen het hy die begeleiding grotendeels behartig nie, maar het hy ook by my 'n waardering gekweek vir die elegansie van teorie. Met genoë kyk ek terug na ons lang en interessante gesprekke oor SVMs en die insig wat ek gekry het uit sy breë vlak van vakkennis. Ek hoop dat ek in die komende jare nog baie uit sy bron van kennis kan put. Verder het ek ons urelange debatte oor wêreld politiek en maatskaplik-ekonomiese probleme baie geniet.

Prof.dr. Klaus-Robert Müller, Prof.dr. Emile Aarts, Prof.dr. Jaap Wessels, en die ander lede van die promosiekommissie wil ek bedank vir hulle betrokkenheid, opmerkinge en bereidwilligheid om mee te werk aan hierdie ondersoek. Aan Dr. Arthur Kordon, Dr. Jaap den Doelder, en Dr. Randy Pell van The Dow Chemical Company wil ek baie dankie sê vir hulle belangstelling in my werk, hulle ondersteuning en waardevolle kommentaar. Arthur, baie dankie vir al die voorstelle om die sagteware te verbeter en vir die opsporing van ontwykende programmeerfoute.

Ek wil ook al my kollega's by Dow-PTC 2 in Terneuzen bedank vir hul vriendelikheid en ondersteuning, asook hulle geduld om my Nederlands te verbeter. Baie dankie dat julle op Vrydae my die geleentheid gegee het om Afrikaans, my moedertaal, te kon praat. Ek hoop dat ons die gewoonte in die komende jare kan voortsit.

Dan wil ek my familie en vriende, Nederlanders sowel as Suid-Afrikanners, uit die diepte van my hart bedank. Sonder julle sou ek nooit so ver as Europa gekom het nie, of dit ooit so lank hier kon volhou nie. In die besonder wil ek my ouers, Bert en Martie Jordaan, bedank vir alles wat hulle vir my gedoen het. Hulle het my gestimuleer om my Godgegewe verstand en talente te gebruik, en ook in elke besluit wat ek ooit geneem het ten volle ondersteun.

Ek weet dat die kennis en verstand waaroor ek beskik net tydelik is. Enige dag kan my Skepper dit van my wegneem. Daarom streef ek daarna om veral 'n waardevolle mens te wees.

Curriculum Vitae

PERSONAL DETAILS

Name: Elsa Jordaan
Date of birth: June 15, 1974
Place of birth: Groblersdal, South Africa.

SCHOOL EDUCATION

1981-1987: Toitskraal Primary School, Marble Hall, South Africa.
1988-1992: Piet Potgieter High School, Potgietersus, South Africa.

UNIVERSITY EDUCATION

1993-1995: Potchefstroom University for Christian Higher Education (CHE),
South Africa
Applied Mathematics and Chemistry (Bachelor's degree)

1996: Potchefstroom University for CHE, South Africa
Applied Mathematics (Honours degree)

1997: Potchefstroom University for CHE, South Africa
Department of Mathematics and Applied Mathematics
Temporary teaching position
Follow courses in preparation for study abroad

1997-1999: Eindhoven University of Technology, The Netherlands
Stan Ackermans Institute
Postgraduate course: Mathematics for Industry
(master's degree)

1999-2002: Eindhoven University of Technology, The Netherlands
Department of Mathematics and Computing Science
Research assistant,
work performed at Dow Benelux B.V. in Terneuzen