# Copyright statement

**BOURNEMOUTH UNIVERSITY**

# On robust and adaptive soft sensors

by

Petr Kadlec

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

School of Design, Engineering & Computing
Bournemouth University

in collaboration with
Evonik Industries AG

December 17, 2009

# Abstract

In process industries, there is a great demand for additional process information such as the product quality level or the exact process state estimation. At the same time, there is a large amount of process data like temperatures, pressures, etc. measured and stored every moment. This data is mainly measured for process control and monitoring purposes but its potential reaches far beyond these applications. The task of soft sensors is the maximal exploitation of this potential by extracting and transforming the latent information from the data into more useful process knowledge. Theoretically, achieving this goal should be straightforward since the process data as well as the tools for soft sensor development in the form of computational learning methods, are both readily available. However, contrary to this evidence, there are still several obstacles which prevent soft sensors from broader application in the process industry. The identification of the sources of these obstacles and proposing a concept for dealing with them is the general purpose of this work.

The proposed solution addressing the issues of current soft sensors is a conceptual architecture for the development of robust and adaptive soft sensing algorithms. The architecture reflects the results of two review studies that were conducted during this project. The first one focuses on the process industry aspects of soft sensor development and application. The main conclusions of this study are that soft sensor development is currently being done in a non-systematic, ad-hoc way which results in a large amount of manual work needed for their development and maintenance. It is also found that a large part of the issues can be related to the process data upon which the soft sensors are built. The second review study dealt with the same topic but this time it was biased towards the machine learning viewpoint. The review focused on the identification of machine learning tools, which support the goals of this work. The machine learning concepts which are considered are: (i) general regression techniques for building of soft sensors; (ii) ensemble methods; (iii) local learning; (iv) meta-learning; and (v) concept drift detection and handling. The proposed architecture arranges the above techniques into a three-level hierarchy, where the actual prediction-making models operate at the bottom level. Their predictions are flexibly merged by applying ensemble methods at the next higher level. Finally from the top level, the underlying algorithm is managed by means of meta-learning methods. The architecture has a modular structure that allows new pre-processing, predictive or adaptation methods to be plugged in. Another important property of the architecture is that each of the levels can be equipped with adaptation mechanisms, which aim at prolonging the lifetime of the resulting soft sensors.

The relevance of the architecture is demonstrated by means of a complex soft sensing algorithm, which can be seen as its instance. This algorithm provides mechanisms for autonomous selection of data pre-processing and predictive methods and their parameters. It also includes five different adaptation mechanisms, some of which can be applied on a sample-by-sample basis without any requirement to store the on-line data. Other, more complex ones are started only on-demand if the performance of the soft sensor drops below a defined level.

The actual soft sensors are built by applying the soft sensing algorithm to three industrial data sets. The different application scenarios aim at the analysis of the fulfilment of the defined goals. It is shown that the soft sensors are able to follow changes in dynamic environment and keep a stable performance level by exploiting the implemented adaptation mechanisms. It is also demonstrated that, although the algorithm is rather complex, it can be applied to develop simple and transparent soft sensors. In another experiment, the soft sensors are built without any manual model selection or parameter tuning, which demonstrates the ability of the algorithm to reduce the effort required for soft sensor development. However, if desirable, the algorithm is at the same time very flexible and provides a number of parameters that can be manually optimised. Evidence of the ability of the algorithm to deploy soft sensors with minimal training data and as such to provide the possibility to save the time consuming and costly training data collection is also given in this work.

# Contents

# List of Figures

# List of Tables

## Acknowledgements

In the first instance I would like to express my gratitude to my supervisor Prof. Bogdan Gabrys. I would like to thank Prof. Gabrys for every single of the countless hours spent discussing this work in its broadest sense. I am also very thankful for the critical and yet supportive feedback I have received as well as for pushing me through the 'lows' of the project.

My thanks go also to the industrial partner of this project, Evonik Industries AG. Without their initiative and support this work would not be possible. In person, much gratitude goes to Monika Berendsen, Reinhard Dudda, Dr. Sibylle Strandt and Dr. Uwe Tanger. In particular, I want to express many thanks to Monika and Reinhard for the preparation of our regular progress meetings and for making the meetings very enjoyable by treating me so well.

Dr. Roman Rosipal contributed to this work by commenting the technical content of parts of this thesis, which is also gratefully acknowledged.

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

I would also like to thank all the people who made the time during which I was working on this thesis so enjoyable. This includes in the first instance my girlfriend Sachiko who was always here to support me. Others who contributed are my former housemates and other PhD researchers from across the whole university.

Finally, as special thanks, I dedicate this thesis to my parents and my brother for their support and believing in me.

# Author's declaration

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

# Nomenclature and conventions

| Symbols | Explanation | Example |
|---|---|---|
| $a$ | Scalar value | $a = 10$ |
| $a_i$ | Scalar value, indexed as part of a vector | $\mathbf{a} = [a_1, \ldots]$ |
| $a_{i,j}$ | Scalar value, indexed as part of a matrix | $A = \begin{bmatrix} a_{1,1} & , \ldots \\ a_{2,1} & , \ldots \end{bmatrix}$ |
| $\mathbf{a}$ | Vector | $\mathbf{a} = [1, 2, \ldots]$ |
| $\mathbf{a}_i$ | Vector, indexed as part of a matrix | $A = [\mathbf{x}_1, \ldots]^T$ |
| A | Matrix | see above |
| $\mathcal{S}$ | Set | $\mathcal{S} = \{(X, \mathbf{y})\} = \{(\mathbf{x}_i, y)\}_{i=1}^{n}$ |

| Reserved characters | Explanation | Example |
|---|---|---|
| $\mathbf{x}$ | Input data instance | $\mathbf{x} = [0.1, 0.3 \ldots]$ |
| $\mathbf{x}_i$ | $i$-th input data instance | |
| $\mathbf{x}_{i,j}$ | $j$-th variable of $i$-th input data instance | |
| $\mathbf{x}_{.,j}$ | $j$-th variable of an input data instance | |
| $\mathbf{y}$ | Output (target) variable | |
| $y^{final}$ | Final predictions of a model, i.e. output of the model | |
| $\mathcal{D}$ | Set of data points | $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ |
| $\mathcal{L}$ | Set of local expert descriptors | $\mathcal{L} = \{(L_{k,j})\}_{k=1;j=1}^{p;m}$ |
| $f(), f(\mathbf{x})$ | Predictor/model | |
| $f_{(k)}$ | $k$-th predictor/model | |
| $\mathcal{F}$ | Ensemble, set of predictors | $\mathcal{F} = \left\{(f_{(k)}\right\}_{k=1}^{p}$ |
| $\mathcal{C}^p$ | Set of predictions and target values | $\mathcal{C}^p = \left\{(f_{(k)}(\mathbf{x})_i, y_i)\right\}_{i=1;k=1}^{n;p}$ |
| $n$ | Number of data instances/samples | |
| $m$ | Number of data variables/features/measurements | |
| $l$ | Number of variables after PCA transformation | |
| $p$ | Number of predictors | |
| $q$ | Number of combinations | |
| $r$ | Number of receptive fields | |
| $\mu$ | Expected value of Gaussian kernel function | |
| $\sigma$ | Variance of Gaussian kernel function | |
| $\Sigma$ | Co-variance matrix | |
| $w, v$ | Combination weights | |
| $med()$ | Median value of a variable | |
| $c(\mathcal{F})$ | Ensemble combination function | |

# List of abbreviations

| | | | |
|---|---|---|---|
| **Soft sensing and pre-processing algorithms** | | | |
| ANN | Artificial Neural Networks | NLPCA | Non-Linear Principle Component Analysis |
| EKF | Enhanced Kalman filter | NNPLS | Neural Network Partial Least Squares |
| EWPLS | Exponentially Weighted Partial Least Squares | PCA | Principle Component Analysis |
| FC | Fuzzy Combinational | PCR | Principle Component Regression |
| FMWPCA | Fast Moving Window Principle Component Analysis | PLS | Partial Least Squares |
| FPM | First Principle Model | PSO | Particle Swarm Optimization |
| GP | Genetic Programming | RBFN | Radial Basis Function Network |
| kNN | k-Nearest Neighbours | RLMS | Robust Least Means Squares |
| LMS | Least Mean Square | RNN | Recurrent Neural Network |
| LTGANN | Learnt Topology Artificial Neural Network | RPCA | Recursive Principle Component Analysis |
| LWR | Locally Weighted Regression | RobPCA | Robust Principle Component Analysis |
| LWPR | Locally Weighted Projection Regression | RTGANN | Random Topology Artificial Neural Network |
| LSSVM | Least Squares Support Vector Machine | SOM | Self-Organizing Network |
| MBPCA | Model-Based Principle Component Analysis | SRM | Stepwise Regression Method |
| MLP | Multi-Layer Perceptron | SVM | Support Vector Machines |
| MPLS | Multi-way Partial Least squares | SVR | Support Vector Regression |
| MLR | Multiple linear regression | TLPCA | time lagged Principle Component Analysis |
| NFS | Neuro-Fuzzy System | TS | Takagi and Sugeno model |
| **Architecture related** | | | |
| CLM | Computational Learning Method | LCU | Local Control Unit |
| CLMP | Computational Learning Methods Pool | MLL | Meta-Level Learning |
| CP | Computational Path | PC | Path Combination |
| FS | Feature Selection | PP | Pre-Processing |
| GPE | Global Performance Evaluation | PPM | Pre-processing Method |
| IS | Instance Selection | PPMP | Pre-processing Methods Pool |
| ISM | Instance Selection Module | RF | Receptive Field |
| **General** | | | |
| FIFO | First In First Out | MSE | Mean Squared Error |
| FPM | First Principle Model | OP | On-line Prediction |
| LASSA | Local Adaptive Soft Sensing Algorithm | PIMS | Process Management System |
| LE | Local Expert | PFD | Process Fault Detection |
| LEC | Local Expert Candidate | PM | Process Monitoring |
| MAD | Median Absolute Deviation | ROSS | Robust On-line Soft Sensor |
| MDM | Model-Driven Model | SFD | Sensor Fault Detection |

## List of publications resulting from this work

The following list contains peer-reviewed conference and journal publications resulting from this work:

- Kadlec, P., Gabrys, B. and Strandt, S., 2009. Data-driven Soft Sensor in the process industry. *Computers and Chemical Engineering*, 33(4), 795-814.
- Kadlec, P. and Gabrys, B., 2009. Architecture for development of adaptive on-line prediction models. *Memetic Computing*, 1(4), 241-269.
- Kadlec, P. and Gabrys, B., 2008. "Gating Artificial Neural Network Based Soft Sensor." *In:* Nguyen, N. T. and Katarzyniak R., eds. *New Challenges in Applied Intelligence Technologies.*, Springer, 193-202.
- Kadlec, P. and Gabrys, B., 2008. "Application of computational intelligence techniques to process industry problems." *In:* Nguyen, N.T., Kolaczek, G. and Gabrys, B., eds. *Knowledge Processing and Reasoning for Information Society.*, EXIT Publishing House, 305-322.
- Kadlec, P. and Gabrys, B., 2007. Nature-inspired adaptive architecture for soft sensor modelling. *NiSIS'2007 Symposium: 3rd European Symposium on Nature-inspired Smart Information Systems*, St Julian's, Malta.
- Kadlec, P. and Gabrys, B., 2008. Adaptive local learning soft sensor for inferential control support. *International Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA)*, Vienna, Austria. IEEE Computer Society, 243-248
- Kadlec, P. and Gabrys, B., 2008. Learnt Topology Gating Artificial Neural Network. *International Joint Conference on Neural Networks (IJCNN)*, Hong Kong, China. IEEE, 2604-2611
- Kadlec, P. and Gabrys, B., 2008. Soft Sensor based on adaptive local learning. *International Conference On Neural Information Processing (ICONIP)*, Auckland, New Zealand. Springer: Lecture Notes in Computer Science, 1172-1179
- Kadlec, P. and Gabrys, B., 2009. Evolving on-line prediction model dealing with industrial data sets. *IEEE Symposium Series on Computational Intelligence 2009 (SSCI)*, Nashville, USA. IEEE, 24-31
- Kadlec, P. and Gabrys, B., 2009. Soft sensors: Where are we and what are the future challenges? *2nd IFAC International Conference on Intelligent Control Systems and Signal Processing (ICONS'2009)*, Istanbul, Turkey.

# Chapter 1

# Introduction

Operational excellence of processing plants has become, more than ever, important for achieving economic and environmental targets in the process industry. In general, operational excellence is a continuous pursuit for improving the processes. As such, it leads to higher cost efficiency, better plant capacity exploitation, and loss reduction as well as achieving compliance with environmental and safety regulations achieved by operating the processes in their optimal states.

Another ongoing change in the process industry is a large instrumentation and digitalisation of the processing plants, which leads to vast amounts of data being recorded and stored. Although this data has now made its way into the low-level control and monitoring of the processing plants, the latent information available in this data provides possibilities going far beyond these applications. However, exploiting this hidden information is not easy and it requires advanced tools to be achieved. One of such tools are *soft sensors*, the central topic of this thesis. Soft sensors have the potential to start a new generation of operational excellence at a relatively low cost because the data needed for their development is already available, as are the required techniques.

This chapter is a general introduction to the topic. It provides a brief overview of the soft sensor types and the techniques that are today being used for their development. Another part of this chapter is the discussion of the issues of current soft sensors that prohibit them from wider applicability in the process industry. The last section of this chapter provides an outline of the thesis, together with descriptions of the main contributions of each chapter.

## 1.1 Introduction to soft sensors

Industrial processing plants are usually heavily instrumented with a large number of sensors. The primary purpose of the sensors is to deliver data for process monitoring and control. However, approximately two decades ago researchers started to make use of the large amounts of data being measured and stored in the process industry by building predictive models based on this data. In the context of the process industry, these predictive models are called *soft sensors*. This term is a combination of the words *software*, because the models are computer programs, and *sensors*, because the models deliver information similar to their hardware counterparts. Other common terms for predictive sensors in the process industry are *inferential sensors* [89, 146] and *virtual on-line analysers*, as they are called in the Six-Sigma context [75].

At a general level, one can distinguish two different classes of soft sensors, namely the model-driven and data-driven type (see Figure 1.1). Although there are some model-driven soft sensors based on the extended Kalman filter [194] or adaptive observers [6] (see for example [31, 90]), this family of soft sensors is most commonly based on First Principle Models (FPM) [142]. First

Principle Models describe the physical and chemical background of the process. These models are developed primarily for the planning and design of the processing plants, and therefore usually focus on the description of the ideal states of the processes. This is only one of their drawbacks, which makes it difficult to base practical soft sensors on them. As a solution, the data-driven soft sensors increasingly gained in popularity.

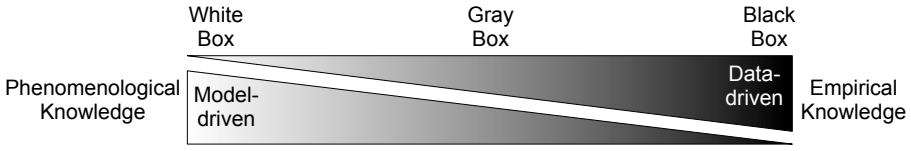



Figure 1.1: Model vs. data-driven soft sensors [55]

The most popular modelling techniques applied to data-driven soft sensors are the Principal Component Analysis (PCA) [88] in a combination with a regression model (Principal Component Regression - PCR), Partial Least Squares (PLS) [198], Artificial Neural Network (ANN) [13], Neuro-Fuzzy System (NFS) [86] and Support Vector Machine (SVM) [173].

The range of tasks fulfilled by soft sensors is broad. The most dominant application area of soft sensors is the prediction of process variables that can be determined either at low sampling rates or through off-line analysis only. Because these variables are often related to the process output quality, or other critical aspects of the process, they are very important for the process control and management. For these reasons, it is of great interest to deliver additional information about these variables at a higher sampling rate and/or at lower financial cost, which is exactly the role of soft sensors. The supervised learning methods applied to these kind of soft sensors are either statistical or computational learning approaches. This soft sensor application field is further referred to as *on-line prediction*. Other important application fields of soft sensors are those of *process monitoring and process fault detection*. These tasks refer to the detection of the state of the process and, in the case of a deviation from the normal conditions, to identification of the deviation source. Traditionally, the process state is monitored by process operators in the control rooms of the processing plants. The observation and interpretation of the process state is often based on univariate statistics and it is up to the experience of the process operator to put the particular variables into relation and to make decisions about the process state. The role of process monitoring soft sensors is, based on the historical data, to build multivariate features that are relevant for the description of the process state. By presenting the predicted process state or the multivariate features, the soft sensor can support the process operators and allow them to make faster, better and more objective decisions. Process monitoring soft sensors are usually based on the PCA and Self Organizing Maps (SOMs) [106]. Figure 1.2 shows the discussed application fields of data-driven sensors and the most commonly applied techniques for each of the applications.

Furthermore, processing plants often embody large numbers of various sensors, therefore there is a certain probability that a sensor occasionally fails. Detection of this failure is the next application area of soft sensors. In more general terms, this application field can be described as *sensor fault detection and reconstruction*. Once a faulty sensor is detected and identified, it can be either reconstructed or the hardware sensor can be replaced by a soft sensor, which is trained to act as a *back-up soft sensor* of the hardware measuring device. If the back-up sensor proves to be an adequate replacement of the physical sensor, this idea can be driven even further and the soft

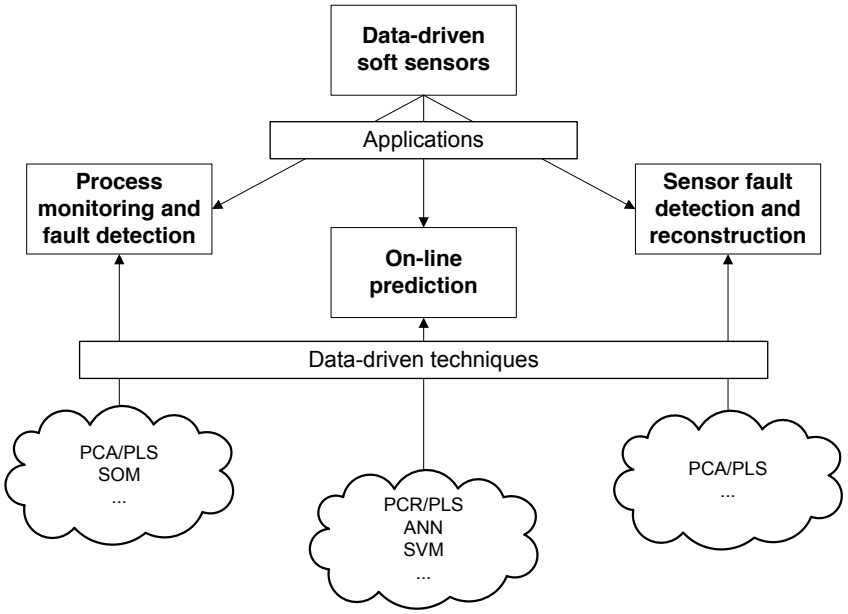



Figure 1.2: Overview of application areas of data-driven soft sensors

sensor can also replace the measuring device in normal working conditions.

## 1.2 Issues of current soft sensors

Despite all the previously listed soft sensor application fields and the high number of publications dealing with their various applications (see Appendix A), there are still some unaddressed issues of soft sensor development and maintenance. To a large extent it is these issues that prohibit the wider spread of practical soft sensor applications in the process industry.

Large portion of the issues can be attributed to the data upon which the soft sensors are built. This data often shows varying quality, which makes the application of common predictive methods such as PCR or MLP difficult. The most common effects, which can be observed in the process industry data, are:

- Data outliers
- Missing values
- Measurement noise
- Data co-linearity
- Drifting data [1].

Currently, the most common approach to deal with these issues is obtaining as much process knowledge as possible and incorporating this knowledge into the model in the form of data preprocessing and model selection. The process knowledge is usually applied to select important variables, to select steady states of the data, to remove data outliers, etc. However, this is problematic since the process knowledge differs from one process to another and as such has to be

[1] In this work the term *drift* is used for any type of change, i.e. gradual as well as abrupt.

acquired for each new soft sensor to be built. After acquiring the knowledge, it has to be manually implemented into the models, which is also very time consuming. It is not difficult to demonstrate that this kind of labour intensive approach is very expensive and thus a significant obstacle to the wider application of soft sensors. The approach chosen in this work is to deal with the issues by equipping the soft sensors with a certain *robustness* towards the above mentioned issues. In this sense, robustness is understood as the insensitivity of the models to data impurities. This is a slightly different notion compared to the one used in robust statistics [82].

Another problematic fact for practical soft sensor applications is related to their run-time maintenance. After the successful launch of the soft sensor, one can often observe a gradual deterioration of its performance. The changes in the data, which cause the performance to deteriorate, are the effect of such phenomena as varying quality of the input raw materials, changes in the catalyst activity, abrasion of mechanical components, external environment changes or changes of the operational state of the process. Usually after some time, the performance of the model reaches an unacceptable level and the model has to be retrained or, in the worst case, rebuilt from scratch.

## 1.3 Project goals

The overall goal of this thesis is the development of a concept which effectively deals with the issues of current soft sensors discussed above. The concept should support:

1. Making the soft sensor development as simple, automated and cheap as possible. This includes providing the possibility to develop the soft sensors off-the-shelf with minimal amount of required process knowledge as well as demand for parameter and model selection;

2. Prolonging the lifetime of the soft sensors developed according to the concept by means of their self-adaptation and effective exploitation of the provided feedback data;

3. Offering high flexibility, i.e. the algorithm can be manually tuned for the specific modelling task if required;

4. Providing mechanisms for the incorporation of the potentially available process knowledge. Despite the fact that the primary goal of this work is to minimise the demand for process knowledge, very often a certain amount of this knowledge is available and therefore it should be possible to effectively apply the available knowledge.

Applying such a concept should, on one hand, significantly simplify the soft sensor development process and, on the other hand, minimise the requirement for periodical checking and tuning of the models.

Of the various types of soft sensors discussed in Section 1.1, this work focuses on the *data-driven on-line prediction soft sensors for continuous processing plants*. In particular, the decision to focus on soft sensors *for continuous processes* was made in consensus with the industrial partner of the project, whose main revenue comes from this type of processes. Another fact that supported this decision is that *on-line prediction* is the most common soft sensor application type. The focus on the research of *data-driven* soft sensors originates in the fact that, in contrast to the model-driven type, this kind of soft sensor allows the project goals like the minimisation of required process knowledge to be achieved.

In order to achieve the stated goals, the following aspects of soft sensor development and maintenance have to be researched and critically analysed:

- The typical soft sensor development methodology
- The data used for the development
- Common pre-processing and modelling techniques for soft sensing
- Soft sensor maintenance practices.

After this analysis, the next goal of the project is to review relevant machine learning techniques and tools, which, based on the critical review of the state-of-the-art of soft sensor development, promise the achievement of the set goals. The studied techniques are:

- Ensemble methods
- Local learning
- Meta-learning
- Concept drift detection and handling.

After the analysis the findings are projected into a concept for building robust and adaptive soft sensors. The proposed concept is also referred to as the *architecture* for the development of soft sensing algorithms, as shown in Figure 1.3. The figure also shows that the next step towards a practical soft sensor is building a *soft sensing algorithm* which is an instance of the architecture. The final step is the training of the algorithm using the data set from the process for which the soft sensor has to be developed.


Figure 1.3: The pathway from the abstract architecture to practical soft sensors followed in this work

## 1.4 Organisation of the thesis

Figure 1.4 shows the structure of the thesis and the main dependencies between the particular chapters.

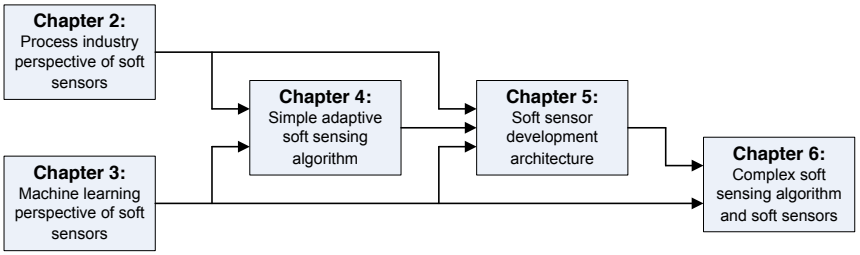



Figure 1.4: The organisation of the chapters of this thesis

Chapter 2 provides a comprehensive introduction to soft sensors in the process industry. The chapter starts with the discussion of the environment in which the soft sensors are being applied.

This is followed by a critical analysis of the characteristics of the data used for soft sensor development and operation. The result of the analysis is that the data often has properties that are harmful for the soft sensor development and operation. Based on the findings and discussions with experienced soft sensor developers, a typical development methodology is provided in this chapter, which is followed by a comprehensive review of the different application areas.

In Chapter 3 soft sensors are reviewed from the perspective of machine learning. First, a theoretical framework, in which the rest of this work is embedded, is briefly described. This is followed by the discussion of the advantages and drawbacks of the most common methods for soft sensor development. Focusing on empirical modelling and keeping in mind the findings from Chapter 2, the techniques which can potentially be useful for dealing with the issues of current soft sensors are also reviewed.

Chapter 4 builds upon the findings from the two preceding chapters and proposes a novel algorithm for on-line prediction soft sensing. This algorithm is based on local learning, ensemble methods and concept drift handling. It is in this chapter where the algorithm is presented and evaluated. The evaluation is performed on real-life process industry data sets. The results of the experiments were promising and encouraged further study of this approach.

Chapter 5 is the most significant step towards the concept for development of robust and adaptive soft sensors. This concept applies the approach from the previous chapter as its core principle and further generalises it into a universal architecture for the development of soft sensing algorithms.

Chapter 6 aims at proving the concept proposed in the previous chapter by means of presenting a soft sensing algorithm which is developed by following the principles defined by the architecture proposed in the previous chapter. The core of the instance is the algorithm proposed in Chapter 4. In the significantly extended form, the algorithm supports the deployment of multiple local experts per receptive field as well as several adaptation mechanisms at different hierarchy levels.

In chapter 7 the work is concluded and the most significant findings of this thesis are highlighted and discussed. Possible directions for further research are also outlined in this chapter.

# Chapter 2

# Process industry perspective of soft sensors

## 2.1 Introduction

This chapter introduces soft sensors from the process industry viewpoint. Soft sensors could be seen as plain predictive models which are trained with the available set of historical data and then deployed to make predictions for the incoming data. However, in reality their development and maintenance are significantly more challenging. To identify the challenges and to select an appropriate strategy to tackle these, it is important to understand the specific aspects of the process industry and their implications for soft sensor development. In order to achieve this goal, this chapter starts with an introduction to chemical industry processes in Section 2.2. This is followed by a discussion of the characteristics of the data collected in the process industry in Section 2.3. Understanding the process data is critical because this work deals with empirical, i.e. data-driven, soft sensors. Another important part of this chapter is a taxonomy of different soft sensor types and application areas presented in Section 2.4. This is followed by the discussion of the state-of-the-art soft sensor development methodology, which shows the way soft sensors are developed in the process industry nowadays. The presented methodology is an outcome of the discussions with experienced soft sensor developers from the process industry and as such shows the critical points of the development from their viewpoint. This chapter also provides a comprehensive review of soft sensor applications across various fields of the process industry. The purpose of the review is to identify: (i) the most popular techniques; (ii) current trends; and (iii) existing gaps in knowledge in state-of-the-art soft sensing. The structure and dependencies between the sections of this chapter are shown in Figure 2.1.

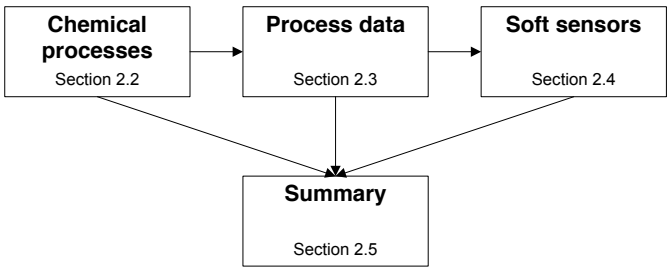



Figure 2.1: The structure of this chapter

## 2.2 Processes in the chemical industry

The task of chemical process plants is to produce a desired product out of a set of input materials. Usually, the materials have to be processed in multiple steps in order to be converted into the final product with the required quality. Figure 2.2 shows an example of a simple process.

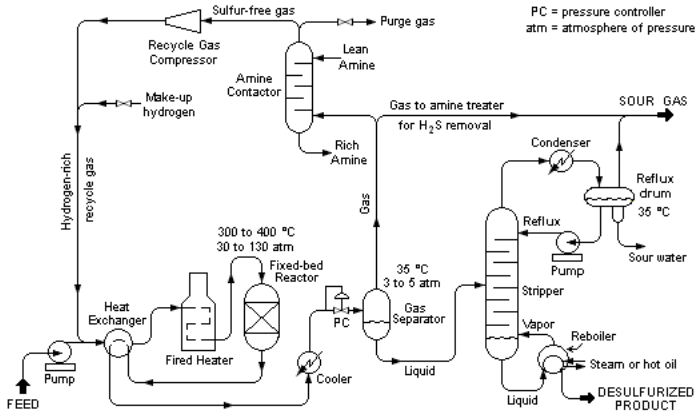



Figure 2.2: Example of a chemical industry process (from content.answers.com)

Processing plants consist of different types of equipment. From the process point of view, the reactor and distillation columns are the most important parts. In the reactor, the actual chemical reactions take place. For this reason the reactors are usually equipped with a large number of sensors. During the reactions the input materials are transformed into the required substances, i.e. products, and inevitably some by-products. At a high level of abstraction, the reactions are described by following type of chemical formulae:

$$A + B + C \rightleftharpoons E + F \tag{2.1}$$

$$F + B \rightarrow G, \tag{2.2}$$

where $A, B, C$ are the input substances, $E$ is the desired product and $F$ is a by-product which may make another reaction for instance with the input $B$ and produce another by-product $G$.

The output of the reactor is usually fed into distillation columns. The purpose of these is to split the desired product from the reaction by-products. In the previous case it would mean the splitting of the product $E$ from the by-products $F$ and $G$. In order to achieve desired product quality, the distillation has to be done in multiple steps by feeding the output of the distillation back to the column or by using a cascade of columns. To be able to control the process effectively, and thus to achieve the required output quality, the reaction conditions and the process state must be monitored and controlled very precisely. Most commonly, the measured values are temperatures, pressures, viscosities, and material flows. In the process industry, the data is not only used for process control and monitoring purposes, but is also stored in large databases called Process Information Management Systems (PIMS).

Starting in the late 1990's, the large amounts of historical data stored in PIMS has found another application in the form of soft sensors (or inferential sensors [90], predictive models, intelligent sensors [107]). This new application arose from the increasing availability and popularity of diverse data-driven computational techniques. As already mentioned, the process variables used for building soft sensors are the sensory data from the plants. Furthermore, one can distinguish

two different types of processing plants in the chemical industry, namely *continuous* and *batch* processing plants.

### 2.2.1 Continuous processes

Continuous processing plants are, as their name suggests, running in an uninterrupted, continuous way. After the start-up phase the plants are operated in a more or less constant and, hopefully, optimal state. As the process should stay in this optimal state most of the time, the soft sensors applied to continuous processes usually focus on the description of this steady-state and are not able to deal with any transient states like the process start-up and shut-down phases. Nonetheless, even the steady-state is progressively changing with time, which has a negative effect on the prediction quality of the soft sensor. The most common causes of the process operating point changes are the changes of the process product demand, the change of the catalyst activity, clogging of heat exchangers, etc.

As continuous processes generate the majority of revenue for most of the process industry companies, this work is biased towards this type of processes.

### 2.2.2 Batch processes

Batch, semi-batch or discontinuous processes (further referred to as *batch* processes only) are processes with a definite duration. Very often these processes are started *on demand* for the production of required product amount. Many processes in the food and biochemistry industry, like the fermentation processes, are of this type. Another field where batch processes are very common is speciality chemistry. Here, special chemicals have to be produced infrequently and often in very small amounts and thus it would not be economical to run the plants in continuous mode.

In [14], it was commented: *Batch processes are experiencing a renaissance as products-on-demand and first-to-market strategies impel the need for flexible and specialised production methods*. This statement is clearly demonstrating the increased demand for modelling tools based on batch process data.

In terms of data-driven modelling, there is a difference between continuous and batch processes. Batch process modelling has to deal with an additional discrete dimension of the data, namely the batch-to-batch variation [132]. While modelling these processes, one has to take into account the finite and varying duration of the processes, the time variance of the particular batches described by the batch trajectory, the high batch-to-batch variance and the starting conditions of the batches [25]. The techniques applied for modelling and monitoring of batch processes are most commonly multivariate statistical techniques. In the case of batch process monitoring, the most common applied method is the PCA (see Section 2.4.3). There are several batch processes monitoring applications reviewed in Section 2.4.3.

## 2.3 Process data

This section focuses on the analysis of the process data that is used for the development of soft sensors. First, the historical and real-time data and the implications for the soft sensor development, operation, and maintenance are discussed. This is followed by the analysis of data characteristics, which are critical for the model development and performance.

### 2.3.1 Historical vs. real-time data

In a real-life industrial scenario there are two types of data with different characteristics available. These are a batch of *historical data* on one hand, and *real-time data* provided as a stream of samples (i.e. multivariate data points) or small batches of samples on the other hand.

#### Historical data

Usually, when dealing with real-life industrial modelling tasks, there is a set of historical recordings available. In the case of process industry data, the historical recordings describe the behaviour of the process in the past and form the basis for the development of the empirical predictive model. Such data sets have a number of properties highly relevant for the model building process. In the case of supervised learning the historical data consists of a number of input variables, i.e. process measurements, and one or more target variables. There is often a relevant delay between taking the measurement, i.e. sampling the input variables, and providing the corresponding target value, i.e. sampling the target variable. This is caused by the fact that the target variables have to be obtained in a time consuming manner (e.g. manual evaluation of the process product quality in a chemical laboratory). Nevertheless, it can be assumed that the delays are compensated by entering the target values at the time point of taking the sample for the historical data.

Furthermore, although the sampling rate of the input data is usually higher than the one of the target variable, in most cases only the labelled samples (i.e. data points containing the target values) are recorded within the historical data. Therefore, the sampling rate of the input variables and the target variable can be assumed as equal.

Another relevant and possibly harming property of the data are the delays between the input variables themselves. Without prior process knowledge these kind of delays are difficult to compensate for and have to be dealt with by the application of an appropriate feature selection algorithm.

The main application of the historical data is for model training, selection and validation. In an adaptive scenario, the historical data set is also used for the adjustment of the parameters of the adaptation techniques.

#### Real-time data

Once the initial model building phase is finished, the model is applied in the on-line operation and needs to deal with the real-time data stream. In comparison to historical data, the real-time data have slightly different characteristics.

It is arriving in an incremental way, i.e. one sample (or a small batch of samples) after another. In general the sampling rate between the input and the target data can differ.

The correct target values, which usually arrive at a lower sampling rate than the input samples and often with certain delays, can be used to evaluate the model performance during the on-line prediction phase. If there is a notable deterioration of the model performance, an adaptation of the model needs be performed using the target data.

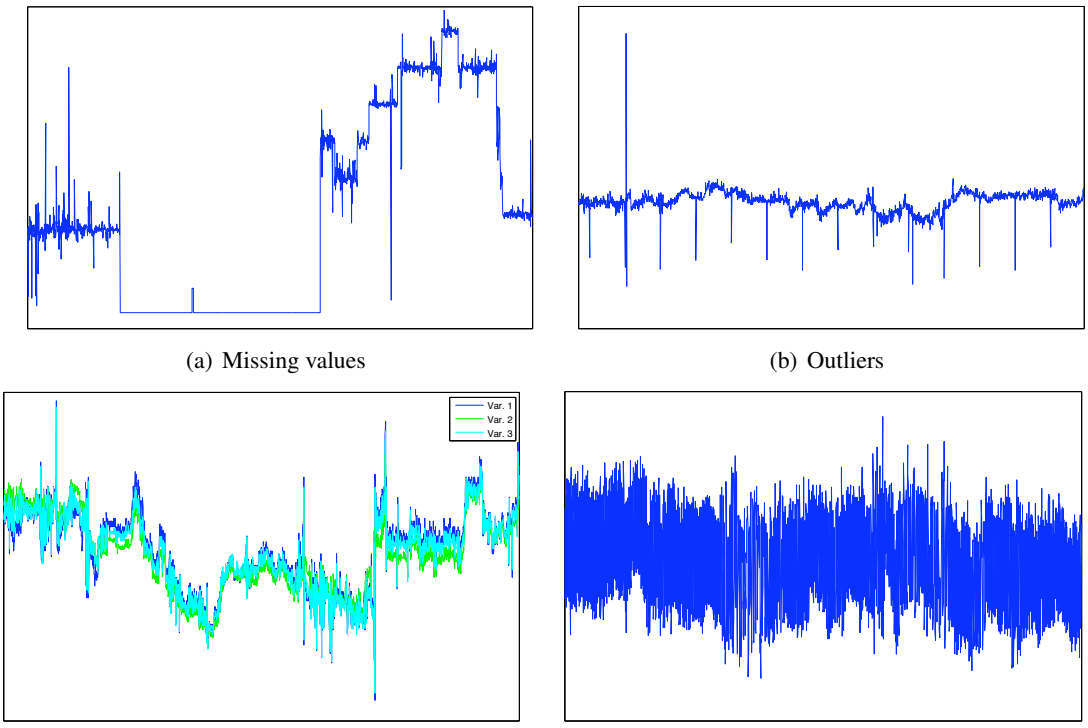
Table 2.1 provides a summary of the two types of data, which are available in the case of realistic industrial data modelling.

### 2.3.2 Process data issues

Figure 2.3 shows examples of variables affected by common issues of industrial data that are going to be discussed further in this section.

| | Historical data | Real-time data |
|---|---|---|
| Purpose | Model training and validation, parameter optimisation | real-life simulation, model adaptation |
| Mode | Batch | Incremental |
| Input variables delays | Possibly present | Possibly present |
| Target variable delays | Compensated | Possibly present |
| Input vs. target sampling rate | Equal | Possibly lower |

Table 2.1: Characteristics of the historical and the real-time data




(a) Missing values (b) Outliers

(c) Data co-linearity (d) Measurement noise

Figure 2.3: Common issues found in industrial data sets

**Missing values:**

Missing data are single samples or consequent sets of them, where one or more variables (i.e. measurements) have a value which does not reflect the real state of the physical measured quantity. The affected variables usually have values like $\pm\infty$, 0.

Missing values in the context of the process industry have various causes. The most common ones are the failure of a hardware sensor, its maintenance or removal. As already mentioned, processing plants are heavily instrumented for the process control purposes, therefore the recorded process data also consists of a large number of diverse variables. In such a scenario, there is a certain probability that some of the sensors will occasionally fail. One should keep in mind that some of the sensor types are mechanical devices (e.g. flow rate sensors) and thus suffer from abrasion effects. Other possible causes of missing data are related to the transmission of the data between the sensors and the database, errors in the database, problems in accessing the database,

etc.

Since most of the techniques applied to data-driven soft sensing cannot deal with missing data, a strategy for their replacement usually has to be implemented. An approach, which is very primitive and not recommended but still commonly applied in practical scenarios, is to replace the missing values with the mean values of the affected variable. Another non-optimal approach is to skip the data samples consisting of variable or variables with the missing values, i.e. *case deletion* [159]. A more efficient approach to missing values handling takes into account the multivariate statistics of the data and thus makes the reconstruction of the missing values dependent on the other available variables of the affected samples ( e.g. maximum-likelihood multivariate approach to missing values replacement [183]). These kind of approaches are related to *sensor fault detection and reconstruction* (for some practical algorithms see Section 2.4.3). From another point of view, one can distinguish two different approaches for dealing with missing values [159]. These are: (i) single imputation where the missing values are replaced in a single step (using e.g. mean/median values); and (ii) multiple imputation, which are iterative techniques where several imputation steps are performed.

A study dealing with missing data was presented in [157]. In this study the authors also discuss two common approaches to handle missing data based on maximum-likelihood and Bayesian multiple imputation.

A two-part publication dealing with multiple imputation techniques for missing values handling was presented in [182, 183]. The first part focuses on the influence of the missing data handling techniques on methods typically applied in chemometrics, i.e. PCR/PLS, etc., whereas the second one proposes a maximum-likelihood-based algorithm for dealing with missing data.

**Data outliers:**

Outliers are sensor values that deviate from the typical, or sometimes also meaningful, ranges of the measured values. One can distinguish between two types of outliers, namely *obvious outliers* and *non-obvious outliers* [144]. Obvious outliers are those values that violate the physical or technological limitations. For example the absolute pressure may not reach negative values or the flow sensor may not deliver values that exceed the technological limitations of the sensor. To be able to detect this type of outlier efficiently, the system has to be provided with the limiting values in the form of a priori information. As for non-obvious outliers, these are even harder to identify because they do not violate any limitations but still do not reflect the correct variable states.

Outlier detection as part of the data pre-processing remains very critical for the soft sensor development because undetected outliers have a negative effect on the performance of the models. For example, the influence of a single outlier can be critical for the PCA [181, 165, 162] (see Section 3.3.1 for more details). Another problem of outlier detection is that even when applying automatic outlier handling pre-processing steps, usually the results have to be validated manually by the model developer. The goal of the manual inspection is to detect any possible outlier *maskings* (i.e. false negative detections; undetected outliers) and outlier *swamping* (i.e. false positive detections; correct values labelled as outliers) [136].

Typical approaches to outlier detection are based on the statistics of the historical data. The most simple approach is the $3\sigma$ outlier detection algorithm (see e.g. [118, 137]), which is based on univariate observations of the variable distributions. This method labels all data samples out of the range $\mu(x) \pm 3\sigma(x)$, where $\mu(x)$ is the mean value and $\sigma(x)$ the standard deviation of the variable $x$, as outliers. A more robust version of this approach is the Hampel identifier [36], which uses a more outlier resistant median and Median Absolute Deviation (MAD) values [136, 137] to

calculate the limits.

The influence of outliers on the identification of linear and non-linear models is discussed in [136]. For the handled models, the Hampel identifier is found to be an effective approach for dealing with outliers. In [128], a moving window filter is combined with the Hampel identifier to obtain an outlier detection and removal system. In contrast to the univariate approaches, the multivariate methods use combinations of more features to detect the outliers. An example from this group based on the PCA is the Jolliffe parameter [88, 191]. A two-stage outlier detection approach is discussed in [72]. The first stage is the application of the PCA, after this the Hotelling's $T^2$ measure [80] can be used to detect outlier candidates that are located outside of the $99\%$ confidance ellipse. These candidates are then further analysed in the second step, where Scheffé's test [70] is applied to these points.

Another, rather general, review of the outlier detection problem and several outlier detection algorithms is presented in [79].

**Drifting data:**

There are two types of drifting data and, dependent on the cause of the drifts, one can distinguish between process and sensor drifts. The causes of the process drift are the changes of the process or of some external process conditions. The processing plants consist of a large number of mechanical elements that undergo steady abrasion during the operation of the plant. This may have an effect on the process itself, e.g. the flow between two parts of the process can decrease due to the abrasion of mechanical pumps. Another cause of the drifting data can also be external influences like changing environmental conditions (e.g. weather influence), the purity of the input materials, catalyst deactivation, etc. These factors have not only an influence on the data but affect the process state as well. Therefore the drifts should be recognised, reported and appropriate actions should be taken to remove their source. This is different in the case of sensor drifts, which are caused by changes in the measuring devices and not by the process itself. The critical point is that this type of drifts, while still observed in the measured data, does not reflect any changes in the process. Therefore in the case of sensor drifts, the action to be taken should be the re-calibration of the measurement devices or the adaptation of the soft sensor without performing any corrective actions to the process.

In terms of the effects on the process data, one can observe changes in the means and variances of the single variables as well as changes of the correlation structure of the data [117].

Distinguishing between the two different drift causes discussed is challenging and once again a lot of expert knowledge is needed in order to take appropriate action. Another challenging aspect of dealing with drifting data is the fact that the changes may progress very slowly and may influence each other, and thus have a non-linear form.

The most common approach to deal with dynamics in the data is to apply the moving window technique. In this case, the model is updated on a periodical basis using only a defined number of the most recent samples. Some examples of the application of this technique in the context of soft sensor modelling were published in [188, 206, 145, 37]. Further approaches for soft sensor adaptation are discussed in Section 2.4.2.

The problems with drifting data are not unique to the process industry data and they can be found in several other fields dealing with changing environments. In the machine learning terminology these problems are summarised under the term *concept drift*, which is discussed in Section 3.7 in more detail.

**Data co-linearity:**

Another challenging issue for soft sensing is related to the structure of the data. Typically, the data measured in the process industry are strongly co-linear. This results from the partial redundancy in the sensor arrangement, e.g. two neighbouring temperature sensors will deliver strongly correlated measurements. At this place it should be recalled that the primary purpose of the data collected within the processing plants is for the process control. For this purpose it is necessary to have detailed information about all process components, which results in a large number of measurements. Such environments are often called *data rich but information poor* [42]. However, for soft sensing the requirements are different, in this case only informative variables are required. Anything else is unnecessarily increasing the model complexity, which often has a negative effect on the model training and performance.

There are two ways to deal with the co-linearity problem. One way is by transforming the input variables into a new reduced space with less co-linearity as it is done in the case of the PCA and PLS. These two approaches are the most popular ones to deal with data co-linearity in the process industry. Examples of applications, where PCA is used, are: [118, 3, 186, 206] and for the PLS [125, 205, 203]. Another way to handle co-linearity is to select a subset of input variables that is less co-linear. These approaches are summarised under the umbrella of *variable (or feature) selection* methods in the computational learning research. A general review of these methods is presented in [74]. Some feature selection methods in the context of soft sensing are also discussed in [191]. Among the discussed approaches are correlation- and partial correlation-based feature selection as well as Mallows' $C_p$ statistics.

**Sampling rates and measurement delays:**

Various sensors usually work at different sampling rates and thus one has to take care to synchronize them. The synchronization of the data is usually handled by the PIMS, which records new data samples only if one of the observed variables changes more than a pre-defined threshold value. The definition of such a threshold is another critical point, which influences the quality of the historical data. Too low value causes the recording of an unnecessarily large number of samples, whereas too high threshold leads to the missing of important process changes. Soft sensing is often applied in multi-rate systems with several operating sampling rates. Such a scenario occurs in a system where some of the variables, usually critical for the process control, are evaluated in laboratories at a much lower sampling rate than the rest of the automatically measured data. This fact causes problems for the modelling and control of the processes. A summary of the last 50 years of multi-rate research is provided in [41].

Additional issues of the process data are the process-related delays in the measurements. The materials in the processes usually have a given run-time through the process (e.g. the dwell period within a reactor or distillation column) and thus it is not reasonable to relate two different measurements taken at the same time at different locations within the process. Instead of this, the delays in the particular measurements should be compensated by synchronizing the variables. However, in order to perform the synchronisation, an extensive knowledge about the process is required.

In the case of batch processes a particular problem is that the different runs of batch processes can have different run times. To be able to apply data-driven methods to batch process historical data, the data must have the same length (i.e. the same number of samples) and thus also require synchronisation.

**Measurement noise:**

Measurement noise is another common effect observed in process industry data. Most of the approaches to soft sensor development are trying to cope with measurement noise during the pre-processing stage of the data processing. This is achieved mainly by applying a smoothing (averaging) filter as a pre-processing step.

The PCA is also a useful tool for dealing with measurement noise. As a least mean squares-based method it can deal with measurement noise as long as it can be assumed as Gaussian noise, i.e. randomly distributed with zero mean value [118]. In the same work the authors also highlight the application of robust statistics, i.e. using the *median* instead of the *mean* operator and MAD instead of the standard deviation, for the normalisation of noisy data.

Zamprogna et al. have shown the robustness of the Partial Least Squares (PLS) method towards measurement noise in [203]. The authors have shown that there are only small changes of the prediction error of a PLS soft sensor with increasing noise levels. The explanation of this fact is that the noise influences mainly the higher-order latent variables that are normally skipped in practical application.

## 2.4 Soft sensors

This section deals with soft sensors in a detailed way. After distinguishing two types of them in Section 2.4.1, a discussion of a state-of-the-art soft sensor development methodology is given in Section 2.4.2. Section 2.4.3 provides a comprehensive overview of published soft sensor application case studies.

### 2.4.1 Model-driven and data-driven soft sensors

At a very general level, one can distinguish two types of soft sensors, namely *model-* and *data-driven* soft sensors. Model-driven models are also called *white-box* models because they have full phenomenological knowledge about the process. In contrast to this, purely data-driven models are called *black-box* techniques because the model itself has no knowledge about the process and is based on its empirical observations only. In between the two extremes there are many combinations of these two major types of models possible. A typical example of such a combination is a model-driven soft sensor making use of a data-driven method for the modelling of fractions that cannot be predicted easily in terms of phenomenological models. These models are called *hybrid* or *grey-box* models.

Model-driven models (MDM), or more specifically, First Principles Models (FPM), are primarily developed for the purpose of planning and development of the process plants. These models are based on equations describing the chemical and physical principles underlying the process. A typical example uses mass-preservation principles, exothermal equation, energy balances, reaction kinetics in the form of reaction rate equations for this purpose. The drawback of these type of models is that their development requires a lot of expert knowledge. This knowledge is not always available. For example, for biochemical processes there is often not enough phenomenological knowledge for accurate description of the processes at hand. Another problem is that the models often describe a simplified theoretical background of the process rather than the real-life conditions of the process, which is influenced by many factors out of the scope of the MDM. Additionally, the model-driven models usually focus on the description of the steady-state of the process and are thus not suitable for the description of any transient states. Nonetheless, model-driven soft sensors

are popular as a support for inferential control. Examples of inferential control applications of first principle soft sensor are [38, 45] where the first example is based on a Kalman filter and the latter one on non-linear observer method. Another example of model-driven soft sensor is [142], where a multi-rate Kalman filter is applied to the control of a polymerisation process.

The focus of this chapter, and of soft sensing in general, is therefore on the data-driven models that have emerged as very attractive modelling approaches enhancing the toolbox of diagnostic, prognostic and decision support methods available for plant operators and embedded in automated control systems.

### 2.4.2 Soft sensor development methodology

This section describes the typical steps and issues of the common practice of soft sensor development. The presented procedure is rather general and can thus be applied for both continuous and batch processes as well as to any of the application areas discussed in Section 2.4.3. An overview of the methodology is presented in Figure 2.4.

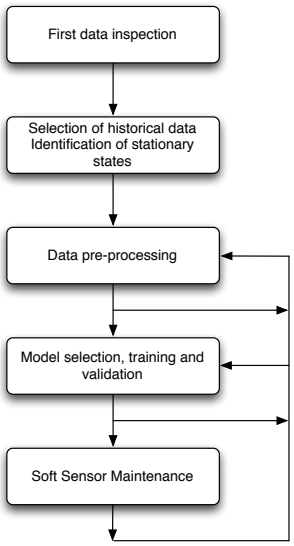



Figure 2.4: A soft sensor development methodology

#### First data inspection

During the initial step, the first inspection of the data is performed. The aim of this step is to gain an overview of the data structure and identify any obvious problems that may be handled at this initial stage (e.g. locked variables having constant value, etc.). The next aim at this stage is to assess the requirements for the model complexity. In the case of an on-line prediction soft sensor, an experienced soft sensor developer can already make a reasonable decision as to whether to use a simple regression model, a rather more complex PCA regression model or a non-linear neural network to build the soft sensor. In some cases, the model family decision at this stage may not be correct, therefore the models and their performance should always be evaluated and compared to alternative models at the later development stages.

Particular attention is paid to the assessment of the target variable. It has to be checked whether there is enough variation in the output variable and if this can be modelled at all.

### Selection of historical data and identification of stationary states

Here, data to be used for the training and evaluation of the model are selected. Next, the stationary parts of the data have to be identified and selected. In the vast majority of cases, further modelling will only deal with the stationary states of the process. The identification of the stationary process states is usually performed by manual annotation of the data.

In [87], the steady-state detection of continuous processes is discussed and a wavelet transform-based approach is applied to perform this task.

In the case of batch processes, there are usually no steady-states and thus the model developer focuses on the selection of representative batch runs rather than on the identification of steady states.

### Data pre-processing

The aim of this step is to transform the data in such a way that it can be more effectively processed by the actual model. An example of a typical pre-processing step is the normalisation of the data to the zero-mean and unit variance (as it is required by the PCA). In the case of the data produced in the process industry, there are several pre-processing steps necessary, which are indicated by the loop around the "Data pre-processing" box in Figure 2.4. The usual steps are handling of missing data, outliers detection and replacement, selection of relevant variables (i.e. feature selection), handling of drifting data and detection of delays between the particular variables. A lot of the listed steps are, at the moment, handled manually or need at least a supervised inspection of the results. The data pre-processing is usually done in an iterative way, e.g. after the standardisation and missing values treatment, which are usually performed only once, an outlier removal and feature selection are repeatedly applied until the model developer considers the data as being ready to be used for the training and evaluation of the actual model. Due to the characteristics of the data discussed in Section 2.3, the importance of the pre-processing is critical.

### Model selection, training and validation

As the model is the engine of the soft sensor, selection of the optimal type is crucial for the its performance. So far, there is no unified theoretical approach for this task and thus the model type and its parameters are often selected in an ad-hoc manner for each case. Model selection is also often subject to a developer's past experience and personal preference, which can be of disadvantage for the final soft sensor. This can be observed among the published soft sensor applications where many of the authors strongly focus on one model type (e.g. PLS) in their field of expertise.

Nevertheless, despite the lack of a common theoretically superior approach to model selection there are a few techniques that can be adopted to this task. A possible approach is to start with a simple model type or structure (e.g. linear regression model) and gradually increase model complexity as long as significant improvement in the model's performance can be observed (using e.g. the Student's t-test [73]). While performing this task, it is important to assess the performance of the model on independent data [193, 77]. The same approach can also be applied to the parameter selection of the pre-processing methods like, for instance, the variable selection.

Additionally, for some industrial processes, it can be difficult to obtain a sufficient amount of historical data for the model development. In such cases it is of advantage to resort to statistical error-estimation techniques, like cross-validation or bagging, discussed in Section 3.4.

After selecting the model structure and training the model, the trained soft sensor has to be

evaluated on independent data once again [193]. There are several tools for the evaluation of the model performance. In the case of numerical performance evaluation the most popular is the Mean Squared Error (MSE), which measures the average square distance between the predicted and the correct value. Another way of performance judgement is using visual representation of the predictions. Among these, the *four-plot analysis* is a useful tool since it provides useful information about the relation between the predictions and the correct values together with analysis of the prediction residuals [55]. A disadvantage of the visual methods is that they require the assistance of the model developer and the final decision, if the model performs adequately, is up to the model developer's subjective judgement.

### Soft sensor maintenance

After developing and deploying the soft sensor, it has to be maintained and tuned on a regular basis. The maintenance is necessary due to the drifts and other changes in the data (see Section 2.3.2) that cause the performance of the soft sensor to deteriorate and have to be compensated for by tuning or re-developing the model.

Currently, most of the soft sensors do not provide any automated mechanisms for their maintenance. This fact, together with the previously discussed evidence of changing data, results in the requirement for manual quality control and maintenance of the soft sensors, which is a significant cost factor for the application of soft sensors. Even worse, there is often no objective measure for assessing the soft sensor quality level and the judgement as to whether a model works well or not is dependent on the model operator is subjective perception based on visual interpretation of the deviation between the correct target value and its prediction.

Nevertheless, there are several adaptive approaches in the literature related to soft sensors. The majority of these approaches are based on adaptive versions of the PCA or PLS, like *moving window PCA* [188] or the *Recursive PCA* [116]. All of these methods rely on periodical or continuous adaptation of the principal component base. Neuro-fuzzy-based soft sensors, such as [122], often provide mechanisms for automatic adaptation. These mechanisms are based on the deployment of new units in the neural structure of the model once a new state of the data is found. An approach related to the neuro-fuzzy methods also providing adaptation possibilities is local learning (see Section 3.5). For example, an adaptive soft sensor developed in local learning framework published in [91] (one of the publications resulting from this work).

Despite the methods for the automated soft sensor adaptation, the model operator still plays an important role as it is his judgment and knowledge of the underlying process that decides the way in which the parameters of the individual adaptation methods are selected (e.g. the length of the window in case of the moving window technique, or a threshold for the deployment of a new receptive field in case of the neuro-fuzzy methods).

### Related methodologies

The discussed methodology, though it is the one most commonly used, is not the only possible way for developing a soft sensor. For example in [191], an alternative methodology for soft sensor, or inferential sensor in Warne's terminology, development has been presented. It is less detailed but still consistent with the methodology presented here. It focuses on three different steps, namely: (i) data collection and conditioning, (ii) influential variable selection and (iii) correlation building. These three steps correspond to the "Selection of historical data," "Data pre-processing" and "Model selection, training and evaluation" in Figure 2.4.

Another work mentioning soft sensor development methodology is [55]. Again, there is no significant difference to the methodology presented in this section.

A rather general methodology for soft sensor development in light of the Six-Sigma process management methodology (for details on Six-Sigma see [163]) was presented in [75].

In [134], in addition to a general three-step soft sensor methodology consisting of (i) process understanding, (ii) data pre-processing and (iii) model determination steps, there is a more specialised methodology for the development of models based on multivariate smoothing procedure discussed. For this methodology, the focus on process knowledge collection and application is evident as the final step deals exactly with this aspect of the development.

### 2.4.3 Soft sensor applications

The applications of soft sensors can be found across many fields of process industry. The most typical examples are the chemical, paper/pulp and steel industry. The following sections list examples of the previously introduced three most common application types of soft sensors across these different fields of the process industry.

#### On-line prediction

The most common application of soft sensors is the prediction of values that cannot be measured on-line using automated measurements. This may be for technological reasons (e.g. there is no equipment available for the required measurement), economic reasons (e.g. the necessary equipment is too expensive), etc. This often applies to critical values that are related to the final product quality. Soft sensors can in such scenarios provide useful information about the values of interest, and in the case when the soft sensor prediction fulfils given standards, it can also be incorporated into the automated control loops of the process. Data-driven soft sensors have been widely used in fermentation, polymerisation and refinery processes.

A comprehensive list of publications dealing with on-line prediction soft sensors can be found in Appendix A.1.

#### Process monitoring and process fault detection

Another application area of soft sensors is process monitoring. Process monitoring can be either an unsupervised learning or binary classification task. The systems can be either trained to describe/analyse the normal operating state or to recognize possible process faults. Commonly, process monitoring techniques are based on multivariate statistical techniques like PCA, or more precisely on Hotelling's $T^2$ [80] and Q-statistics [83]. These measures have, on one hand, the advantage of considering all input features, i.e. using multivariate statistics, and on the other hand, providing information about the contribution of the particular features to a possible violation of the monitoring statistics [30]. Another popular method for process monitoring are SOMs.

Several process monitoring and process fault detection applications of soft sensors are listed in Appendix A.2.

#### Sensor fault detection and reconstruction

The vast majority of modelling techniques applied within the process industry as soft sensors are not able to handle data from faulty sensors as a matter of their normal operation. Therefore there

is a need to identify and replace sensor and process faults before the actual model building and application.

Some publications dealing with sensor fault detection and reconstruction soft sensors are provided in Appendix A.3.

### Soft sensor applications summary

Table 2.2 is a summary of the soft sensors application discussed in this chapter and presented in Appendix A.1 - A.3.

| Publication | Applied method(s) | Applic. type | Process description | Process type |
|---|---|---|---|---|
| [22] | SRM (ARMAX) | OP | Particle size estimation in a grinding plant | Cont. |
| [134] | PCA/PLS+LWR | OP | Toluene composition in a splitter column, diesel temperature in crude oil column | Cont. |
| [91] | MLR ensemble | OP | Thermal oxidiser NOx prediction | Cont. |
| [40] | MLP | OP | Sugar quality estimation | Cont. |
| [90] | MLP, FPM, eKF | OP | Biomass estimation in a fermentation process | Batch |
| [144] | MLP, NNPLS | OP | Refinery process | Batch |
| [127] | MLP | OP | Control loop support of ethanol production support | Batch |
| [134] | MLP+Expert System | OP | Silica content control in the steel production | Cont. |
| [56] | MLP | OP | C4 and C5 Concentration prediction in a debutanizer refinery process | Cont. |
| [39] | MLP, RBFN, SVR | OP | Two simulated biochemical processes | Batch |
| [185] | RBFN | OP | Membrane separation process modelling | Cont. |
| [93] | MLPs ensemble | OP | Industrial drier | Cont. |
| [85] | MLP, RBFN, Hybrid (MLP/RBFN+FPM) | OP | Biomass concentration prediction | Batch |
| [166] | RNN+FPM | OP | Degree-of-cure prediction in epoxy/graphite fiber composites process | Cont. |
| [28] | RNN | OP | Biomass concentration prediction | Batch |
| [29] | RNN | OP | Melt-flow-length prediction in injection molding process | Cont. |
| [202] | RNN | OP | Three simple simulated processes | Cont. |
| [53] | Generalised ANN | OP | Diacetyl concentration prediction | Batch |
| [118] | PCA | OP | Product estimation in cement kiln, $NO_x$ monitoring | Cont. |
| [146] | PCA | OP, SFD | Air emission monitoring | Cont. |
| [204] | PLS, PCA | OP | Simulated distillation column | Batch |
| [37] | EWPLS | OP | Stirred reactor, flotation circuit | Cont. |
| [145] | RPLS | OP | Research octane number prediction in refinery process | Cont. |
| [54] | LSSVM | OP | Gasoline absorbing rate in FCC | Cont. |
| [200] | LSSVM | OP | Light diesel freezing point detection in FCC | Cont. |
| [129] | ANFIS | OP | Rubber viscosity estimation | Cont. |
| [191] | PCA+ANFIS | OP | Polymeric-coated substrate anchorage | Cont. |
| [121] | NFS + GA | OP | Light diesel freezing point detection in FCC | Cont. |
| [4] | NFS | OP | Penicillin production bioprocess | Batch |
| [190] | NFS | OP | Propylene purity prediction in a distillation column | Cont. |
| [122] | Evolving NFS | OP | Crude oil distillation in refinery process | Cont. |
| [43] | NLPCA+NNPLS | OP | $NO_x$ Prediction in exhaust gas | Cont. |

| | | | | |
|---|---|---|---|---|
| [116] | PSO+MLP | OP | Ethylene distillation column | Cont. |
| [97] | Analytic NN+SVM+GP | OP | Interface level estimation in a neutralization unit | Cont. |
| [27] | FPM+RBFN | OP | Microbial population in a bioreactor | Batch |
| [148] | Intelligent soft sensor | OP | Sulphite pulping system | Batch |
| [72] | SRM, TS, FC, PLS, WBM, ANN | OP | Copper concentrate grade in a rougher flotation bank process | Cont. |
| [117] | RPCA | PM | Rapid thermal annealing process | Batch |
| [132] | PCA | PM | Polymerisation process | Batch |
| [151] | MBPCA | PFD | Ethylene compressor | Batch |
| [188] | FMWPCA | PM | Simulated FCC unit process | Cont. |
| [3] | PCA+PLS | PM | Lumber drying | Batch |
| [205] | PLS | OP, PM | Simulated penicillin production process | Batch |
| [78] | FDA | PM | Quadruple tank process; Polyester film manufacturing process | Cont. |
| [2] | SOM | PM, OP | Cont. pulp digester; steel production process; pulp and paper industry | Cont. |
| [201] | FPM+ANN | PFD | FCC reactor | Cont. |
| [100] | PCA, SOM, RBFN | PM, PFD | Ethylene cracking process | Cont. |
| [125] | MPLS | PM | process end point detection | Batch |
| [51] | PCA | PFD, SFD | boiler process | Cont. |
| [114] | TLPCA | PFD, SFD | polymerisation process | Batch |
| [186] | PCA | SFD | centrifugal chiller process | Cont. |
| [187] | PCA | SFD+PM | air handling unit | Cont. |

Table 2.2: List of the reviewed soft sensor publications

The list of soft sensor application examples presented in this work is not exhaustive since the amount of published soft sensor applications is too large to be fully covered. Instead of this, this work focuses on one hand on recent publications and on the other hand on non-traditional approaches.

Assuming the presented examples are a representative sample of the recent soft sensors, the distribution of current soft sensing methods is presented in Figure 2.5. The figure shows the current trend in soft sensing. The most popular methods for soft sensor building are the multivariate statistical techniques, i.e. the PCA and the PLS, which together cover $38\%$ of the applications presented in this review. Other technique commonly applied in soft sensing are the neural network-based methods like MLP, RNN, etc. But some of the most recent applications rely on methods that have been recently finding their way into much broader application areas. These are, for example, the neuro-fuzzy methods, which have the advantage of providing an intrinsic mechanism for adaptation/evolution as well as SVM, which have their justification in the theory of machine learning and have additionally proved to have very good generalisation ability accross a number of different application areas.

### The role of process knowledge

A common point of most of the presented soft sensors is the need for the involvement of process-related knowledge. This can be done in several ways. If we ignore the purely model-driven soft sensors, which are out of the scope of this work, one can distinguish different levels of process information influence. One type of process information involvement is the construction of additional features that describe some process related properties. The hope is that these features will

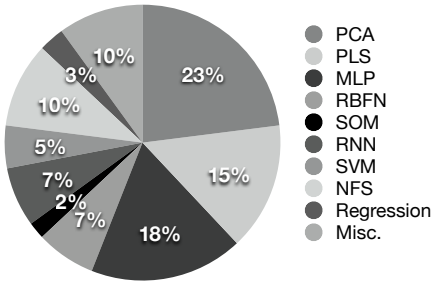



Figure 2.5: Distribution of computational learning methods in soft sensing

be correlated with the modelled target variable and thus have a positive effect on its modelling. Another way of applying process knowledge to data-driven soft sensing is during the initial modelling steps (see Section 2.4.2). The pre-processing steps especially require a lot of attention from the model developer, who often has to interview the process experts in order to be able to carry out manual variable selection, to evaluate the results of the particular pre-processing steps, etc.

#### Further reviews of soft sensor applications

There are several other publications providing reviews of soft sensor applications, among them is a review of regression-based models, ANN, PCA, Kalman filter and Expert Systems applications in process industry [71]. Gonzalez focuses in the cited review on the applications aspects of the before mentioned methods and provides a list of application examples.

In [55], apart from extensive handling of soft sensors and their application to process monitoring and control, an overview of applications of mainly ANN-based soft sensors is given.

Dote and Ovaska also provide a list and a discussion of applications of soft computing techniques in the process industry in their general review of industrial application of soft computing methods [44].

Focusing on process fault detection and diagnosis, Venkatasubramanian published an extensive three-part review. The first part provides an introduction to process fault detection and abnormal event management [176]. Apart from a taxonomy of the different approaches, this part presents quantitative model-based methods for process fault detection and criteria that are used to evaluate and compare the different approaches. The second part of the series deals with qualitative model representations and search strategies for process fault detection [174]. These methods are usually based on first principle descriptions of the processes. Finally, the third part focuses on process data-based techniques [175]. These models can be both qualitative, e.g. enhanced Kalman filter models, as well as quantitative, which can be based on any data-driven method. From the data driven approaches, the authors describe PCA/PLS, Statistical Classifier and ANN-based techniques. Additionally, a comparative study of the various techniques presented in the three-part review is given.

## 2.5 Summary

Figure 2.6 provides a summary of this chapter, which covers the main aspects of the soft sensor development from the process industry point of view.

The main purpose of this chapter is the review of current soft sensing practises and the identification of challenges and issues of current soft sensor development and maintenance practises.

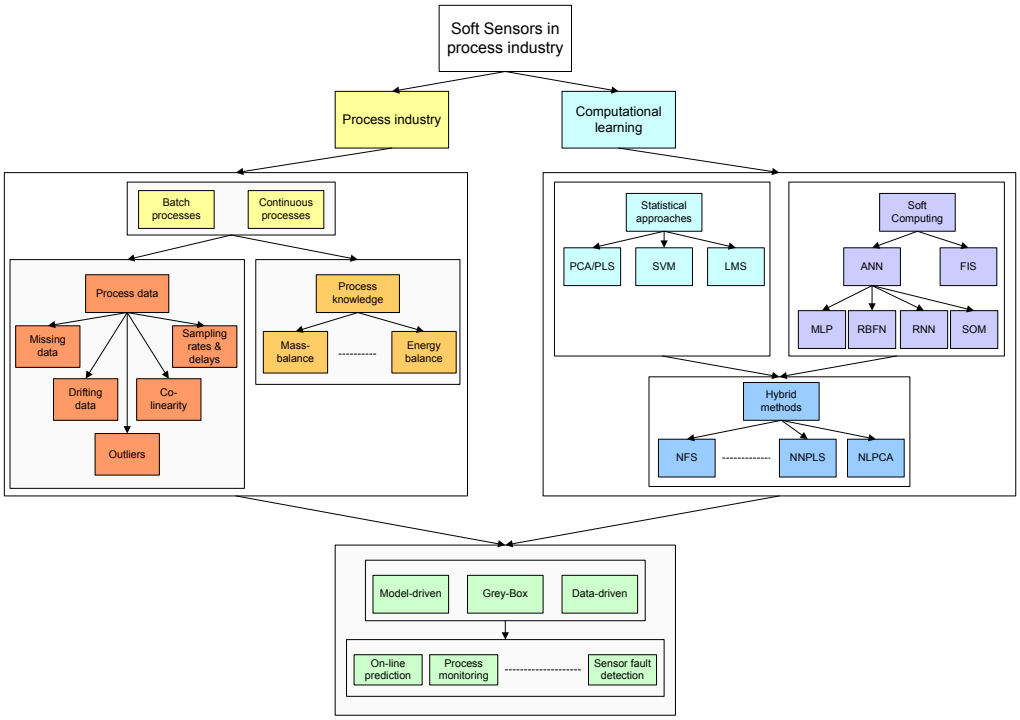



Figure 2.6: Soft sensors as combination of process industry data and computational learning tools

Currently, a lot of effort has to be spent on manual pre-processing of the data as well as on the model selection and validation steps during the soft sensor development phase. In order to be able to deal with data impurities, which can be commonly found in the process industry data, a lot of knowledge about the underlying process has to be collected and implemented into the models.

Another costly issue identified is related to model maintenance. After the successful launch of the soft sensor, one can often observe a gradual deterioration of its performance. The decrease of the predictive performance is usually caused by gradual changes in the process, by changes in the operating state of the process, by changes in the input materials or by hardware sensor abrasion. This shows that trying to compensate for the changes by means of incorporation of process knowledge is almost impossible and the soft sensor should, ideally, be adaptive and try to compensate for the changes by exploiting some adaptive and evolving mechanisms. Currently, the latter approach is the common practice. The evidence for this fact can be found in the list of the soft sensors provided in this chapter. From the soft sensors reviewed here, only a small fraction show an ability to adapt or evolve with the changing environment.

The next chapter reviews machine learning techniques that promise to help tackle the issues identified in this chapter. In particular the focus is set on methods that are potentially useful for automated dealing with data pre-processing, model selection and validation as well as soft sensor adaptation.

# Chapter 3

# Machine learning perspective of soft sensors

## 3.1 Introduction

While the previous chapter focused on the introduction to soft sensing from the process industry point of view, in this chapter the same topic is examined from the machine learning viewpoint. Machine learning provides the toolbox that can be used for the development and automated maintenance of soft sensors.

Before discussing the particular techniques, relevant theoretical aspects are discussed in Section 3.2. This is followed, in Section 3.3, by a detailed presentation of the most common on-line prediction soft sensing techniques identified in the previous chapter.

The following four sections introduce more advanced machine learning concepts, which are going to be applied later on in this work in order to deal with the issues in current soft sensor development. The discussed concepts include ensemble methods (Section 3.4), local learning (Section 3.5), meta-learning (Section 3.6) and concept drift detection and handling (Section 3.7). The structure and dependencies between the sections of this chapter are shown in Figure 3.1.

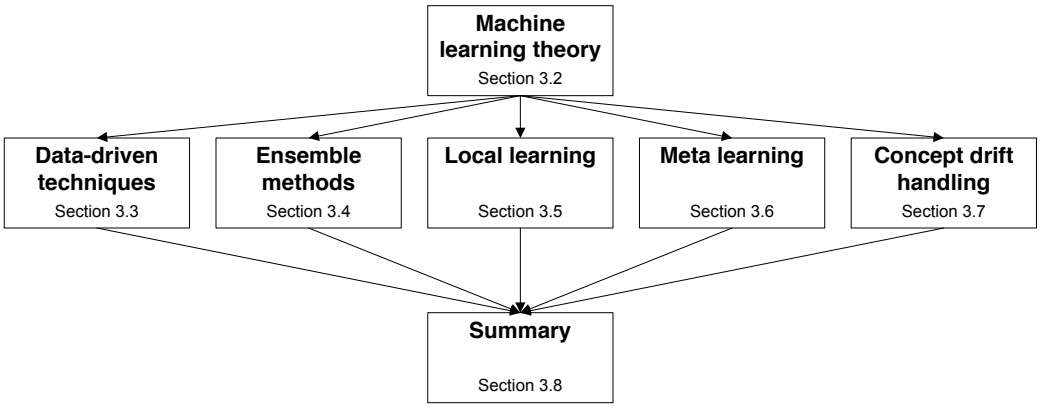



Figure 3.1: The structure of this chapter

## 3.2 Theoretical framework

In this section, soft sensors are embedded into a theoretical machine learning framework. From the several possible types of soft sensors, this thesis concentrates on the on-line prediction soft sensors (see Section 2.4.3) that, from the machine learning point of view, represent supervised predictive models with continuous target variables.

The prerequisite for supervised learning is a training data set $\mathcal{S}^{train}$, which, in the process industry, is usually retrieved from the process information system and is referred to as the *historical data* (see Section 2.3.1). This data set has the following characteristics:

$$\begin{aligned} \mathcal{S}^{train} &= \left\{\left(X^{train}, Y^{train}\right)\right\} \quad \text{with} \\ X^{train} &= (\mathbf{x}_1, \ldots, \mathbf{x}_n)^T \quad \text{and} \\ Y^{train} &= (\mathbf{y}_1, \ldots, \mathbf{y}_n)^T, \end{aligned} \tag{3.1}$$

where $X^{train} \in \mathbb{R}^{m \times n}$ are the $n$ input samples (data points) organised in a matrix, with each row vector $\mathbf{x}_i$ corresponding to one input sample that further consists of $m$ measurement variables (or features), i.e $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,m})$. Each of the input samples $\mathbf{x}_i$ can be also interpreted as a point in the $m$-dimensional input space $\mathcal{X}$. In general, for every input sample, there is a q-dimensional target value $\mathbf{y}_i \in \mathbb{R}^q$ assigned. However, in this work, the target space is restricted to a one dimensional output space $\mathcal{Y}$, i.e. $y_i \in \mathbb{R}^1$.

The target variable is generated by a hidden function $\phi$, called the *target function*, which maps the input space onto the output space:

$$\phi : \mathcal{X} \rightarrow \mathcal{Y}. \tag{3.2}$$

The target function is unknown and it is the task of the learning algorithm $L$ to find an approximation to this function given the historical data set $\mathcal{S}^{train}$ and a (randomly initialised) predictor function $f^{init}$:

$$f^{trained} \leftarrow L(\mathcal{S}^{train}, f^{init}). \tag{3.3}$$

The predictor can be, for example, a simple regression model, a principal component regression model with a given number of principal components, or a multi-layer perceptron with a given number of hidden units and randomly initialised weights (for several possibilities for the predictor functions see Section 3.3). The outcome of the learning process is the trained predictor $f^{trained}$. This predictor is further referred to as the *model*. The model is able to map the input samples to the output space, i.e. $y_i^{pred} = f^{trained}(\mathbf{x}_i)$ and approximates the target function $\phi$. Going back to one of the previous examples, for the linear regression function the trained predictor is a vector of weights $f^{trained} := \beta = [\beta_0, \beta_1, \ldots, \beta_m]$, which can be used to calculate the predicted target value: $y_i^{pred} = \beta_0 + \sum_{j=1}^{m} \beta_j x_{i,j}$.

In order to be able to assess and rank the outcomes of the learning process, an error function $e(f, y)$ is required. This function allows the calculation of the distance between the correct target value $y$ and the value predicted by the predictor function, $f^{trained}$, given an input sample $\mathbf{x}$. In the case of noisy observations:

$$y = \phi(\mathbf{x}) + \epsilon, \tag{3.4}$$

where $\epsilon$ is assumed to be a normally distributed random variable with zero mean value, it can be shown that the optimal error function (in the maximum likelihood sense), which eliminates the

influence of the random noise, is the Mean Squared Error (MSE) (see e.g. [13, p. 195],[18]):

$$e(f(X), \mathbf{y}) = \frac{1}{n} \sum_{i=1}^{n} \left(f(\mathbf{x}_i) - y_i\right)^2, \tag{3.5}$$

where $n$ is the number of data samples. Having the error function, the learning process can be described as the search for an optimal prediction function $f^{optimal}$ which minimises the error function $e()$:

$$f^{optimal} = \underset{f}{\operatorname{argmin}}\, e(f(X), \mathbf{y}). \tag{3.6}$$

However, in practical scenarios, such as soft sensor development, the model developer is interested in finding a model that performs optimally for future samples, which are not available at the time of the model development. This performance is called the *generalisation performance*. Since it is not possible to calculate the performance on future data samples it has to be estimated from the available training data. There are several ways to estimate the generalisation performance. The easiest way, though not the most accurate, is the *hold-out* estimation, where the training data $\mathcal{S}^{train}$ is split into an actual training and a validation part. The predictor is first trained on the new training data and then tested on the validation data, which gives an estimation of its generalisation performance. The problem of this approach is that in practical scenarios the size of the training data set is often limited by several factors and applying the previous approach *wastes* the validation data because it cannot be used for model training. As a solution to this impracticality, several approaches summarised under the term *resampling techniques* were developed in the statistics (see also [193] for an overview of resampling methods). The two most common of these methods are:

- k-fold cross-validation: This method cyclically splits the training data $\mathcal{S}^{train}$ into an actual training data and validation data, whereas the size of the splits depends on the number of folds/splits. For the k-fold cross-validation the size of each validation set is $n\frac{1}{k}$ of the size of $\mathcal{S}^{train}$ while the rest of the samples, i.e. $n(1-\frac{1}{k})$, are used as training data. The result is $k$ different training-validation splits and thus k models. The main benefit of this technique is that it guarantees that all of the available samples are used for model training as well as model validation.

- bootstraping: In the case of bootstraping the data is sampled randomly with replacement from the original pool of samples $\mathcal{S}^{train}$, which generates subsamples of the original training data that are used to train the models. The performance of the particular models is estimated by validating them on the remaining, unseen, samples. The performance of the predictor is the average built over the model performances.

### Bias-variance decomposition

The study of the generalisation error led to one of the most important theoretical findings in machine learning, namely to the *bias-variance decomposition* [66]. Geman et al. have shown that in the case of the quadratic error. The generalisation error of a predictor can be split into two components; the (squared) bias and the variance of the error. The formal form of the decomposition

for the quadratic error function, with respect to different training sets of fixed size, is:

$$\begin{aligned} E_{\mathcal{S}}\left[\left(\mathbf{y}^{test} - f_{(k)}(X^{test})\right)^2\right] &= \left(E_{\mathcal{S}}\left[\mathbf{y}^{test} - f_{(k)}(X^{test})\right]\right)^2 + \\ &\quad E_{\mathcal{S}}\left[\left(\mathbf{y}^{test} - E_{\mathcal{S}}\left[f_{(k)}(X^{test})\right]\right)^2\right] \\ &= bias(f_{(k)})^2 + variance(f_{(k)}) \\ &\quad \text{with} \quad f_{(k)} \leftarrow L(\mathcal{S}_i \in \mathcal{S}^{train}, f^{init}) \end{aligned} \tag{3.7}$$

In the previous equation, the expectation value $E_{\mathcal{S}}$ is built over the random training data sets $\mathcal{S}_i$ of constant size, $f(X^{test})$ are the predictions made by the predictors, given the test input data $X^{test}$ and $\mathbf{y}^{test}$ are the correct target values for the calculation of the quadratic error.

The significance of the above decomposition originates from the fact that it splits the generalisation error into two components that balance each other. The bias component describes the expected error over all trained predictors $f_{(k)}$. This component is in general high for simple models, i.e. models with a low number of free parameters, and relatively low for complex models with a high number of degrees of freedom, which on average will be able to fit the target function better. On the other hand, the variance component describes the degree of variability between the predictions of the predictors trained on different training sets. Contrary to the bias, this value is in general low for simple models since, due to a low degree of freedom, all these models will make similar predictions. For the complex models, the variance component is usually high because they tend to *overfit* the training data and then produce erroneous predictions for the test data. This phenomenon is also called *bias-variance dilemma* since finding an optimal predictor means finding a balance between the bias and variance error.

An important implication of the decomposition is that better generalisation performance cannot be achieved by choosing a more complex model structure, e.g. neural networks with a higher number of hidden units since this merely decreases the bias term while potentially increasing the variance term. The goal of model selection is finding an optimal model with balanced bias and variance errors.

The ultimate ambition in machine learning is the search for techniques that reduce both of the error terms at the same time. This can be achieved by increasing the number of training examples, which in turn allows the training of more complex models. However, in practical scenarios where obtaining more training data is often very expensive or even impossible, one needs to find another way of achieving this goal. Two such methods of approaching simultaneous minimisation of bias and variance from different directions are the ensemble methods and local learning discussed in Section 3.4 and Section 3.5 respectively.

### Learning-forgetting dilemma

When dealing with adaptive systems, one inevitably has to deal with the *learning-forgetting dilemma* [48], sometimes also called stability-plasticity dilemma [21]. The goal when dealing with the dilemma relates to finding an optimal trade-off between learning new information and forgetting old information. Adaptive learning systems with generalisation capability need a mechanism for forgetting old information as their capacity is limited and as such they suffer from *negative interference*. This refers to forgetting of past useful knowledge while learning new information [156]. The extreme manifestation of this problem is *catastrophic forgetting* [58]. According to the previously cited work, the problem can be prevented by: (i) having representative validation data set; (ii) memorising all training data samples; or (iii) incorporation of strong prior knowledge.

## 3.3 Data-driven techniques for soft sensing

This section provides a discussion and further references to the most popular techniques for data-driven soft sensing identified in Section 2.4.3.

### 3.3.1 Principal Component Regression

Principal Component Regression (PCR) models consist of the Principal Component Analysis (PCA) as a pre-processing step that is followed by a regression model operating in a reduced space which is the output of the PCA [88].

The PCA algorithm reduces the number of variables by building linear combinations of the input variables in such a way that these combinations cover the highest variance in the input space and are additionally orthogonal to each other. In the context of the process industry data, this is a very useful feature because the process industry data is often co-linear (see Section 2.3.2 for the discussion of co-linearity in process industry data). Another particular useful feature of the PCA is that it supports dealing with noisy data (refer to Section 2.3.2 for more details).

The PCA is often applied in combination with a regression model but it is not necessarily restricted to this type of model and can generally be combined with any predictive modelling technique.

The PCA algorithm proceeds as follows: a normalized (zero mean and unite variance) matrix $X \in \mathbb{R}^{n \times m}$ consisting of $n$ samples and $m$ variables (features) holding the input data can be transformed into $l$-dimensional ($l \leq m$) PCA space $X^{PCA} \in \mathbb{R}^{n \times l}$ in the following way:

$$X^{PCA} = P^T X, \tag{3.8}$$

where $P \in \mathbb{R}^{m \times l}$ is the transformation matrix. There are several ways to calculate the transformation matrix $P$. In the case of the covariance approach, the covariance matrix $C$ of the input data $X$ has to be calculated:

$$C = \frac{1}{n} X^T \cdot X. \tag{3.9}$$

Next the eigenvalues and eigenvectors of this matrix are derived:

$$V^{-1} C V = eig(C), \tag{3.10}$$

where $eig(C)$ is the diagonal eigenvalues matrix and $V$ the eigenvector matrix. The eigenvalues $\lambda_i$ are then sorted in descending order such that $\lambda_1 > \lambda_2 > \ldots > \lambda_m$. The columns of PCA-transformation matrix $P$ are then formed by the eigenvectors $\mathbf{v}_i$ corresponding to the $l$ highest eigenvalues:

$$P = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_l]^T \quad \text{with } \mathbf{v}_i \in \mathbb{R}^m. \tag{3.11}$$

Having the PCA transformed version $X^{PCA}$ of the original input matrix $X$, one can build a multiple regression model using the Least Means Squares (LMS) algorithm:

$$\mathbf{y}_{pred} = X^{PCA} \beta + \epsilon, \tag{3.12}$$

where $\mathbf{y}_{pred}$ is the prediction vector of the LMS model, $\epsilon$ is the residual error vector of the model and $\beta$ are the parameters of the regression model, which are estimated as:

$$\beta = ((X^{PCA})^T X^{PCA})^{-1} (X^{PCA})^T \mathbf{y}, \tag{3.13}$$

where $\mathbf{y}$ are the available target values. Applying the fact that the vectors in $X^{PCA}$ are orthogonal, the previous equation can be simplified to:

$$\beta = ((X^{PCA})^T X^{PCA})^{-1}(X^{PCA})^T \mathbf{y} = L^{-2}(X^{PCA})^T \mathbf{y}, \tag{3.14}$$

where $L$ is another diagonal matrix whose elements are $\sqrt{\lambda_k}$ (see [88, p. 168].

Although the PCA is a well established and powerful algorithm it has several drawbacks and limitations. One of the limitations is that the pure PCA can only effectively handle linear relations of the data and thus cannot deal with non-linearity. Another issue is the selection of the optimal number of principal components. This can be approached by using, for example, the parameter cross-validation technique or any of the methods discussed in [88, p. 173]. Another problem is that the principal components describe the input space very well but do not reflect the relation between the input and the output data space, which actually has to be modelled. A solution to this problem is given by the Partial Least Squares (PLS) [65], which instead of covering the input space variance, pays attention to the covariance matrix that brings together both the input and the output data space.

A practical problem of the PCA, which is relevant for soft sensor development, is its sensitivity to outliers. The estimation of the direction of the maximal variance of the data can be severely affected by outliers. There are two ways of dealing with this issue [35]. The first is by using robust estimates of the covariance matrix on which the PCA relies [34]. However, this type of estimator has been found to have a low breakdown point (percentage of samples which can be effected before the estimation gets corrupted) in high dimensions [35]. The second type of approach to robust PCA is using robust dispersion estimators replacing the original outlier sensitive variance measure. These approaches are referred to as *projection pursuit* methods [115]. A popular robust measure applied in projection pursuit is the Median Absolute Deviation (MAD) measure, which has the following form:

$$MAD(\mathbf{x}_{\cdot,j}) = med_{i_1}(|\mathbf{x}_{i_1,j} - med_{i_2}(\mathbf{x}_{i_2,j})|), \tag{3.15}$$

$\mathbf{x}_{\cdot,j}$ is the $j$-th variable of the samples, and $med$ the median function. This method will be later applied as a pre-processing step in the experiments in Chapter 4 and 6 and referred to as *robust PCA*.

### 3.3.2 Artificial Neural Networks

The original idea of Artificial Neural Networks (ANN) [13] was to build computational models motivated by the operation of biological *neurons* that are the basic information processing units in nervous systems. There is a large variety of computational intelligence models that are more or less biologically plausible and are summarized under the term *artificial neural network*.

The following subsections will focus on two ANN variants that are commonly applied to process industry data modelling.

#### Multi-Layer Perceptron

Multi-Layer Perceptron is a feed-forward ANN as shown in Figure 3.2. It consists of one input layer, one output layer, and at least one hidden layer. The role of the input layer is to collect the input data and provide it to the hidden layer for further processing. The number of units in the input layer is equivalent to the dimensionality of the input data. Each of the input units is

connected to each hidden unit and the connections between the units carry weights. The role of the hidden units is to collect the signals at their input, i.e. the outputs of the preceding layer, multiply them by the connection weights, build a sum of them and process them using the transfer function $g^{hidden}$:

$$x_i^{hidden} = g^{hidden}\left(\sum_j w_{j,i}^{hidden} x_j^{in}\right), \tag{3.16}$$

where $x_j^{in}$ is the $j$-th variable of the input sample, $w_{j,i}^{hidden}$ is the weight between the $j$-th input unit and $i$-th hidden unit and $x_i^{hidden}$ the output of the $i$-th hidden unit. The transfer function can be any differentiable non-linear function. Often used are the sigmoid functions such as:

$$g^{hidden}(x) = \frac{1}{1+\exp(-x)}. \tag{3.17}$$

Once the signals are processed by the hidden layer they are passed further to the next hidden layer, in the case when there is more than one hidden layer, or to the output layer otherwise. The output layer can consists of one or more neurons. In the case of typical regression modelling (such as described in Section 3.2), the output layer consists of only one unit:

$$x^{out} = g^{out}\left(\sum_i w_i^{out} x_i^{hidden}\right), \tag{3.18}$$

with $x^{out}$ being the output of the MLP, $w_i^{out}$ the weight between the $i$-th hidden unit and the output unit and $g^{out}()$ the transfer function of the output neuron.

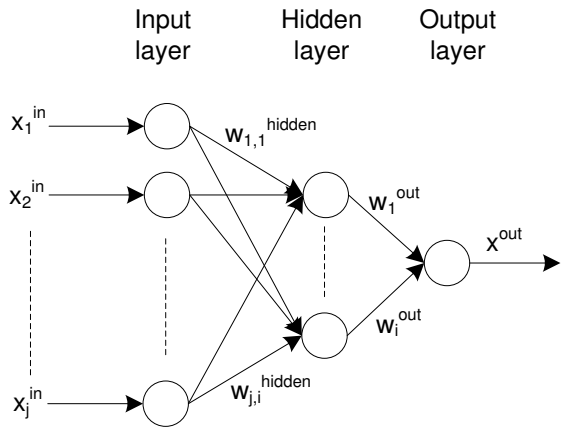



Figure 3.2: Multi-Layer Perceptron structure (without bias connections)

A remarkable theoretical property of the MLP is that they are universal function approximators, which means that, provided enough training data and complex enough structure, they can be trained to approximate any function with any given accuracy [60].

Learning of the MLP (and ANN in general) is achieved by applying an algorithm for finding the optimal weights between the neurons. The most popular of these algorithms is the *back-propagation* algorithm originally proposed in [195].

**Back-propagation algorithm** The breakthrough of ANNs, and especially of MLPs, came in the 1980s after the application of the *back-propagation* algorithm for their training [152]. This algorithm is a generalisation of the *delta rule* [48, p. 283], which was originally applied as a learning rule for linear perceptrons. The original version of the algorithm is a gradient descent method updating the weights in the direction of the steepest descent in the weight-error space:

$$w^{new} = w + \Delta w \qquad \text{with } \Delta w = -\eta \frac{\partial E}{\partial w} \tag{3.19}$$

$$E = \frac{1}{2} \sum_{a=1}^{n} (x_a^{out} - y_a)^2 \tag{3.20}$$

where $\eta$ is the learning rate of the algorithm and $E$ is the quadratic error calculated over the training samples. For the quadratic error function and the derivatives for the output and hidden weights take the following for:

$$\frac{\partial E}{\partial w_i^{out}} = -2(y - x^{out})(g^{out})'(x^{out}) x_i^{hidden} \tag{3.21}$$

$$\frac{\partial E}{\partial w_{j,i}^{hidden}} = -2(y - x^{out})(g^{out})'(x^{out})(g^{hidden})'(x_i^{hidden}) x_j^{in} \tag{3.22}$$

The original version of the algorithm presented in Equation 3.19, also called *vanilla* back-propagation, was later modified and several versions such as gradient descent with momentum or Scaled Conjugate Gradient (SCG) emerged. In the following paragraph the SCG method is discussed as it is the learning method that will be later applied as the MLP learning algorithm.

The SCG algorithm was developed to overcome the problem of the original conjugate gradient descent that relies on the line search algorithm (see [13, p. 272]), which is computationally demanding [130]. The method is a second-order derivative using the Hessian matrix H. The elements of this matrix are second order partial derivatives of the error function, i.e. $H_{i,j} = \frac{\partial^2 E}{\partial w_i \partial w_j}$. The SCG method is an iterative search of the local minimum of the error function. The search starts in the direction of the steepest gradient descent and moves in this direction until a local minimum is found. The second descent direction equals the direction of constant gradient direction but changing magnitude, i.e. the direction in which the direction of the gradient remains constant. The trick of the scaled conjugate descent is that, instead of the evaluation of the full Hessian matrix, it evaluates the Hessian matrix multiplied by the search direction (vector), which is a vector and thus needs fewer evaluations (more details of the algorithm can be found in [13, p. 282]).

**Issues of MLPs** Despite their popularity, there are some issues with the MLPs. The most significant from the practical point of view is that they are prone to get stuck in local minima during the learning process. This can result in sub-optimal performance on the future test data and makes the performance of the trained model strongly dependent on initial conditions like weights initialisation. It will be demonstrated later in Chapter 4 that in the case of industrial data sets this fact can lead to significantly different model performances despite equal parameter settings.

Another problem, which also has a critical influence on the generalisation performance of the trained model, is the difficulty with the estimation of the optimal model topology.

**Radial Basis Function Network**

A Radial Basis Function Network (RBFN) usually consists of three layers, namely an input, hidden and output layer and is, in its topology, similar to the MLP (see Figure 3.2). Nevertheless, two aspects make the RBFN different to MLPs. The first difference is the way the hidden units are activated (i.e. how the weights between the input and output layer are calculated) and the other one is the applied transfer function of the hidden units.

The transfer function of the RBFN hidden units is a Gaussian function that gives this type of feed-forward network its name; *radial basis function*. The Gaussian function has the following parameters, (i) a vector of mean values $\mu$, defining the position of centre of the function in the input space; (ii) and a covariance matrix $\Sigma$, defining the shape of the function. The Gaussian function is defined as:

$$h(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma(\mathbf{x}-\mu)\right). \tag{3.23}$$

The weights between the input and hidden layer are assigned to the values of $\mu_{i,j}$, where $j$ refers to the variable of the input vector $\mathbf{x}^{in}$ and $i$ to the hidden unit, i.e. the Gaussian function. The hidden units process the input by measuring the (usually Euclidean) distance of the input $\mathbf{x}^{in}$ to the centre of the units $\mu_j$, traversing this distance through the Gaussian function and providing the activation of the Gaussian function $x_i^{hidden}$ at their output for further processing:

$$x_i^{hidden} = h(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_i)^T\Sigma_i(\mathbf{x}-\mu_i)\right). \tag{3.24}$$

In the case of regression models, the output unit usually has a linear transfer function and calculates the summation of the hidden units' output weighted by the weights between the hidden and output layer:

$$x^{out} = \sum_i w_i x_i^{hidden}, \tag{3.25}$$

where $w_i$ is the weight between the $i$-th hidden unit and the output unit.

There is a set of parameters that has to be learnt, these are the shape of the Gaussian functions $\Sigma$, their centres $\mu$ (the weights between the input and hidden layer) and the weights between the hidden and output layer. In the simplest scenario, the parameters of the Gaussian functions are learnt off-line by clustering the available data into a given number of cluster and then calculating the Gaussian function parameters for each cluster. Having the Gaussian functions, the weights of the output layer can be calculated for example by applying the Least Mean Squares (LMS) algorithm, i.e.:

$$\mathbf{w} = \left(\left(X^{hidden}\right)^T X^{hidden}\right)^{-1} X^{hidden}\mathbf{y}, \tag{3.26}$$

where $\mathbf{w}$ are the weights between the hidden and output layer, $X^{hidden}$ are the input data samples and $\mathbf{y}$ are the corresponding target values.

### 3.3.3 Support Vector Machines

Due to their theoretical background in the statistical learning theory, Support Vector Machines (SVM) [173] gained the attention of the computational learning community. The general Support Vector Regression (SVR) algorithm is presented in [164, 46] and will be discussed in this section. The first step is constructing a linear function in a high-dimensional space using a kernel function

$\varphi()$:

$$g(x) = \omega^T \varphi(x) + b, \tag{3.27}$$

where $g()$ is the regression function and $\varphi()$ is the kernel function. The goal is to optimise the parameters $\omega$ and $b$ in order to fulfil the following condition:

$$\left| y - \omega^T \varphi(x) - b \right| < \epsilon, \tag{3.28}$$

where $\epsilon$ is the precision of the model predictions. The above optimisation problem can be expressed as:

$$\begin{aligned} \text{minimise} \quad & \frac{1}{2}\omega^T\omega + C\sum_i (\xi_i + \xi_i^*) \\ \text{subject to:} \quad & y_i - \omega^T\varphi(x) - b \leq \epsilon + \xi_i \\ & \omega^T\varphi(x) + b - y_i \leq \epsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0, \end{aligned} \tag{3.29}$$

in these equations, $C$ is a trade-off parameter defining the influence of errors larger than $\epsilon$ on the objectives function and $\xi_i, \xi_i^*$ are called slack variables, which are both zero if the sample is within the $\epsilon$ bound. $\xi_i$ has the value of the difference between the observed value and $\epsilon$, if the sample is above the $\epsilon$ limit. The same is valid for $\xi_i^*$ if the sample is below the bound. This formulation corresponds to the following loss function (called Vapnik's epsilon-insensitive loss):

$$L = \begin{array}{cc} 0 & \text{if } |y - g(x)| \leq \epsilon \\ |y - g(x)| - \epsilon & \text{otherwise} \end{array} \tag{3.30}$$

The above problem can be transformed into the dual formulation using Lagrange operators:

$$\begin{aligned} L := \quad & \tfrac{1}{2}\omega^T\omega + C\textstyle\sum_i (\xi_i + \xi_i^*) - \sum_i (\eta_i\xi_i + \eta_i^*\xi_i^*) \\ & - \textstyle\sum_i \alpha_i(\epsilon + \xi_i - y_i + \omega^T\varphi(x_i) + b) \\ & - \textstyle\sum_i \alpha_i^*(\epsilon + \xi_i^* + y_i - \omega^T\varphi(x_i) - b). \end{aligned} \tag{3.31}$$

The Langrange operators $\eta_i, \eta_i^*, \alpha_i$ and $\alpha_i^*$ are all positive. By setting the partial derivatives of Lagrange function with respect to the primal variables to zero, i.e. $\partial_b L = 0, \partial_\omega L = 0, \partial_{\xi_i^*} L = 0$ the dual optimisation problem can be derived:

$$\begin{aligned} \text{maximise:} \quad & -\frac{1}{2}\sum_{i,j} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(x_i, x_j) \\ & -\epsilon\sum_i (\alpha_i + \alpha_i^*) + \sum_i (y_i(\alpha_i - \alpha_i^*)) \\ \text{subject to} \quad & \sum_i (\alpha_i - \alpha_i^*) = 0 \text{ and } \alpha_i, \alpha_i^* \in [0, C], \end{aligned} \tag{3.32}$$

where $k()$ can be any function fulfilling the Mercer condition, i.e. $k(x_i, x_j) = \varphi(x_i)^T\varphi(x_j)$. After calculating the optimal Lagrange parameters $\alpha_i$ and $\alpha_i^*$, the partial derivative $\partial_\omega = \omega -$

$\sum_i (\alpha_i - \alpha_i^*)\varphi(x_i)$ can be rewritten to:

$$\omega = \sum_i (\alpha_i - \alpha_i^*)x_i, \tag{3.33}$$

which allows the reformulation the original function $g()$ as:

$$g(x) = \sum_i (\alpha_i - \alpha_i^*)k(x_i, x) + b. \tag{3.34}$$

The remaining unknown in the equation is $b$. This can be calculated by applying the Karush-Kuhn-Tucker conditions [164]. The previously cited work can also be referred to find details of the calculation of $b$.

The above optimisation problem has to be solved using quadratic programming techniques. The complexity of the problem is independent of the dimensionality of the input space, however it grows with the number of training samples. Therefore, for large data sets the calculation of the support vector can become computationally infeasible.

### Least Squares Support Vector Machines

The Least Squares Support Vector Machine (LSSVM) [167] algorithm offers a solution to the above complexity problem of the traditional SVM by splitting the global optimisation problem into a set of linear optimisation sub-problems. This is achieved by defining a slightly different objective function (c.t. Equation 3.29):

$$\begin{aligned} \text{minimise} \quad & \frac{1}{2}\omega^T\omega + \gamma\frac{1}{2}\sum_i e_i^2 \\ \text{subject to:} \quad & y_i = \omega^T\varphi(x_i) + b + e_i \end{aligned} \tag{3.35}$$

The difference to the standard SVM is the equality condition in Equation 3.35 (compare to Equation 3.29) and the quadratic error term in Eq. 3.35. The Lagrangian for the objective function has the following form:

$$L := \quad \tfrac{1}{2}\omega^T\omega + \gamma\tfrac{1}{2}\textstyle\sum_i e_i^2 - \sum_i \alpha_i \left(\omega^T\varphi(x_i) + b + e_i - y_i\right), \tag{3.36}$$

where $\alpha_i$ are the Lagrange multipliers. Building the partial derivatives of $L$, i.e. $\partial L_\omega, \partial L_b, \partial L_{e_i}, \partial L_{\alpha_i}$ (see [167] for more details) results in a set of linear equations:

$$\begin{bmatrix} 0 & \mathbf{1}^T \\ \mathbf{1} & k(x,x) + \gamma^{-1}I \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix}. \tag{3.37}$$

Solving the above linear equation provides the parameters $\alpha_k$ and $b$, which allows obtaining the prediction $g(x)$ using the following linear (in the kernel space) equation:

$$g(x) = \sum_i \alpha_k k(x, x_i) + b. \tag{3.38}$$

**Issues of SVM**

A particular issue of SVM is the training process complexity, which grows with the increasing amount of training data. There are also problems with providing adaptation mechanisms for SVM. Nevertheless, there are some publications which focus on this aspect (see e.g. [150]).

## 3.4 Ensemble methods

### 3.4.1 Introduction to ensembles

Ensemble methods are predictive model building techniques that for a given training data build a set, i.e. an ensemble, of models rather than a single model as is the case with the traditional model building. Investigations into ensemble methods have formed a lively research area ever since the appearance of the first works in this field at the beginning of the 1990s, see e.g. [76, 199, 139, 47] for some early works. The first approaches for ensemble methods evolved from statistical re-sampling techniques like jackknifing, bootstrapping and cross-validation. After these initial works, ensemble methods have been studied intensively and proven theoretically [199, 111, 102, 59] and practically [133, 7, 154, 63] to improve the prediction performance of the models.

To build an ensemble, two crucial tasks for the model building can be identified. First, a set of diverse ensemble members (see 3.4.3 for the discussion on diversity in ensemble methods) has to be built and then the models or their predictions have to be combined in order to obtain the final prediction of the ensemble.

**Ensemble members generation:** Ensemble methods that actively influence the generation of the members in order to improve the prediction performance are called *generative ensembles* [172]. The most popular of these methods are:

- Bagging [17]: generates multiple training sets of constant size by re-sampling the training data with replacement, each of the sub-sets is used to train an ensemble member.
- Boosting [158]: the ensemble members are trained iteratively, the training set for each new ensemble member is drawn from the original training data according to a distribution that reflects the prediction error of the previous member.
- Modular neural networks/ Mixture of experts [84] (see Section 3.5.1): in this case, the ensemble members are specialised on regions of the input space, the division of the input space is done by a gating network.

The outcome of this stage is a set of $p$ predictive models:

$$\mathcal{F} = \left\{ f_{(k)} \right\}_{k=1}^{p} \tag{3.39}$$

**Prediction combinations:** This work is restricted to the combination at the prediction level and it does not deal with combinations at the model level because these approaches are algorithm dependent, which goes against the scope of this work. For the combination at the prediction level, there are several approaches that can be applied. Assuming, there is a validation data set consisting of the input data $X^{val}$ and the target variable $\mathbf{y}^{val}$:

$$\mathcal{S}^{val} = \left\{ X^{val}, \mathbf{y}^{val} \right\} = \left\{ (\mathbf{x}_j, y_j) \right\}_{j=1}^{n^{val}}, \tag{3.40}$$

where $n^{val}$ is the number of validation samples, the following approaches to combination are commonly applied:

- best model selection: $f = \text{argmin}_i \, e(f_{(i)}(X^{val}), \mathbf{y}^{val})$.
- model prediction averaging: $f = \frac{1}{p} \sum_i f_{(i)}$ (e.g. in the case of Bagging).
- weighted (convex [1]) combination: $f = \sum_i w_i f_{(i)}$ with $\sum_i w_i = 1, \forall i : w_i > 0$; e.g. Boosting where the weighs are assigned to the prediction accuracy of the ensemble members, or in the case of Mixture of Experts, the weights are assigned according to the output of the gating network.
- combination through a trainable function $g()$ optimised on $\mathcal{S}^{val}$: $f = g(\mathcal{F})$, with $g : \mathcal{F} \rightarrow \mathbf{y}$ (e.g. Stacked generalisation [199]).

In the above equations, $e()$ is an error function (see Section 3.2), $w_i$ are the combination weights and $f$ is the final, i.e. combined, predictor.

The effectiveness of the above combination approaches in the context of neural network predictors was studied in [140], the following paragraphs provide a summary of the findings.

The authors have shown analytically that the selection of the best model is not efficient and that the averaging approach should be preferred. For the averaging approach, the authors demonstrate that the quadratic error of the ensemble $(f(\mathbf{x}) - y)^2$ is decreasing with the increasing size of the ensemble $p$:

$$(f(\mathbf{x}) - y)^2 = \frac{1}{p} \left( \frac{1}{p} \left( \sum_{k=1}^{p} (y - f_{(k)}(\mathbf{x}))^2 \right) \right). \tag{3.41}$$

The previous equation shows that the prediction error of the ensemble can be made arbitrarily low merely by adding new models to the ensemble. This seems to be a very encouraging result but, unfortunately, the equation is valid only under the constrain of statistical independence of the quadratic prediction error of the ensemble members, which is gradually more and more difficult to achieve with the increasing size of the ensemble.

Next, Perrone and Cooper demonstrate that the performance can further be improved using the weighted sum approach. In this case the challenging task is to find the optimal weights for the combination. In the case of convex combination, i.e. $\forall i : 0 \leq w_i \leq 1$ and $\sum_i w_i = 1$ the optimal weights have the following form:

$$w_k = \frac{\sum_l C^{-1}_{k,l}}{\sum_k \sum_l C^{-1}_{k,l}}, \tag{3.42}$$

where $C_{ik}$ are the entries of the correlation matrix of the ensemble members' prediction errors. The difficulty with this approach is the estimation of the correlation matrix and its inverse, as in practical scenarios the rows of $C$ will be linearly dependent (e.g. in case of correlated prediction of two ensemble members), which results in an instability of the matrix inversion.

The last possibility for the combination, namely the functional combination, approach is very general since any learning method can be applied to map the individual predictions onto the final prediction. The other combination methods can be represented as special cases of this general method.

[1] convexity is used to guarantee certain beneficial properties of the combinations, e.g. the existence of a global optimum

### 3.4.2 Bias-variance-covariance decomposition

The bias-variance decomposition discussed in Section 3.2, was extended to ensembles in the form of the *bias-variance-covariance decomposition* in [171]. For a convex combination of estimators, the variance component can be further broken down into the variance and covariance component. In the case of the mean combination of predictors, it can be achieved by replacing the predictor $f$ with the mean combination $\overline{f} = \frac{1}{p}\sum_k f_{(k)}$, which leads to the following form of the variance term:

$$\begin{aligned} variance(\overline{f}) &= E\left[(\overline{f} - E\left[\overline{f}\right])^2\right] = E\left[\left(\frac{1}{p}\sum_k f_{(k)} - E\left[\frac{1}{p}\sum_k f_{(k)}\right]\right)^2\right] \\ &= \frac{1}{p^2}E\left[\sum_k \sum_{l\neq k}\left(f_{(k)} - E\left[f_{(k)}\right]\right)\left(f_{(l)} - E\left[f_{(l)}\right]\right)\right] + \\ &\quad \frac{1}{p^2}E\left[\sum_k \left(f_{(k)} - E\left[f_{(k)}\right]\right)^2\right] \\ &= \frac{1}{p^2}covariance(\overline{f}) + \frac{1}{p^2}variance(\overline{f}). \end{aligned} \tag{3.43}$$

The expectation $E\,[]$ in the above equation is calculated in respect of the different training data sets as well as different initialisations of the predictors. An interesting fact is that, since the covariance part of Equation 3.43 can get negative values, it can decrease the overall generalisation error of the model. It also shows that the quadratic error of the ensemble depends on the bias, the variance and the relationship, i.e. the covariance, of the ensemble members [26].

As shown in Equation 3.43, ensemble methods can be theoretically expected to decrease the quadratic prediction error by decreasing the variance term of the bias-variance decomposition (see Section 3.2). There is also empirical evidence showing that ensemble methods decrease variance [17, 7] or both the variance and bias at the same time [7, 126].

### 3.4.3 The role of diversity in ensembles

The above discussion has shown a benefit for the generalisation performance of ensembles when their members produce uncorrelated errors, which can be seen as an effect of disagreement about the prediction among the ensemble members. The intuitive explanation of this effect is that the ensemble does not benefit from members making the same predictions, i.e. the performance of an ensemble combining ten models predicting the same values is equal to the performance of a single model.

The need for diversity was theoretically shown in the *ambiguity decomposition* in [111]. This decomposition has the following form:

$$(f - y)^2 = \sum_k w_k (f_{(k)} - y)^2 - \sum_k w_k (f_{(k)} - f)^2, \tag{3.44}$$

where $f$ is the combined prediction of the ensemble, $y$ the correct target value, $f_{(k)}$ a prediction of an ensemble member and $w_k$ the combination weights. Equation 3.44 is interesting for several reasons. On one hand it splits the squared prediction error of the ensemble into two independent terms, one describing the average error of the independent ensemble members and the other one,

the *ambiguity term*, describing the diversity level among the ensemble members. On the other hand it shows that the generalisation error of the ensemble is equal to the weighted generalisation error of the particular models less the ambiguity term. This is a strong theoretical justification for the ensemble methods because it shows that, as long as there is an ambiguity among the members, the combined prediction will be lower than the average individual member error. However, the decomposition does not say that there is no ensemble member with lower error than the combination.

Unfortunately, the ambiguity decomposition is also not the *holy grail* of machine learning since there are some obstacles related to it. The major one is that for the minimisation of Equation 3.44, a set of models that are diverse and at the same time maximally accurate is needed. Another challenge is finding the optimal weights for a given set of models.

The ambiguity decomposition can also be set in relation to the bias-variance-covariance decomposition using the Equations 3.44 and 3.43 [20, 19, 26]. After some manipulations, it can be shown that the ambiguity part of Equation 3.44 can be understood as a combination of the ensemble variance and covariance, while the first term of the ambiguity decomposition equals to a sum of the squared bias and variance. The presence of the variance term within both parts of the ambiguity decomposition explains why it is not possible to manipulate the ambiguity without influencing the first term of the decomposition.

A popular solution for finding an ensemble that is optimised under the ambiguity decomposition was presented in the framework of neural networks in the form of the *negative correlation learning* [120]. In this case, there is a penalty term added to the optimised error function. The penalty term is related to the diversity of the output of the ANNs and as such encourages weights that result in uncorrelated error patterns.

In a more general context, there is another way for the estimation of the combination weights optimising the ensemble generalisation error using the ambiguity decomposition presented in [111]. Although there is no analytical solution for the optimal weights, an estimate can be found based on the predicted generalisation performance of the individual ensemble members and the correlation between the models' predictions $C_{kl} = \sum_{\mathbf{x}\in X} f_{(k)}(\mathbf{x}) f_{(l)}(\mathbf{x})$ by optimising the following equation:

$$(f - y)^2 = \sum_k w_k (f_{(k)} - y)^2 + \sum_{k,l} w_k C_{k,l} w_l - \sum_k w_k C_{k,k} \tag{3.45}$$

This equation can be optimised using, for example, linear programming techniques [111].

So far, the requirements for diversity have been identified from practical and theoretical points of view and the influence of the ensemble diversity on the ensemble generalisation error have been shown. An important question to be addressed now is: What are the mechanisms for the creation of diverse model populations? There has been several works dealing with this question. For example, [112, 153] provide a review of diversity generation mechanisms in a classification context. In the context of regression models, Brown et al. proposed a taxonomy for diversity generation mechanisms in [19].

On a high level, the authors distinguish between *implicit* and *explicit* methods for diversity generation. Implicit methods are techniques like bootstraping (see Section 3.4.1), which generate the diversity in a random manner without optimising any function, while explicit methods attempt to manipulate the diversity in a more deterministic way. Explicit, or active, diversity management is performed for example by the boosting method [49], where each new data set covers samples that were not processed correctly by the available predictors. As another way of classifying of diversity creation methods, Brown et al. propose the following types of diversity:

- Starting point in hypothesis space: The starting point in the hypothesis space can be modified, for example, by random initialisation of the weights of ANN. This corresponds to implicit diversity generation mechanism. This method is available for any technique that is prone to local minima.

- Set of accessible hypotheses: This can be varied in two different ways, namely by changing the training data set or by changing the architecture of the learning machine. Some approaches to changing the training data structure were discussed in Section 3.4.1, others are feature selection, instance filtering, methods related to receptive fields building, AdaBoost [59], etc. As for the changing of the architecture of the learning machine, this can be done for example by combining ANN with SVMs or any other predictive method. The intuitive explanation is that different learning approaches will generate different models with different (i.e. uncorrelated) error patterns and consequently by optimising the ambiguity of the ensemble (see Equation 3.44) .

- Traversal of the hypothesis space: This can be achieved, for example, by adding a penalty term to the learning algorithm, which will force the algorithm to move into *unpopulated* areas of the hypothesis space, as done by negative correlation learning in the framework of neural networks.

## 3.5 Local learning

The distinguishing characteristics of local learning methods is that they train a set of models, each of which is trained on limited partition of the data space. Local learning algorithms evolved from the so called *lazy learning* methods [1].

In lazy learning, all training samples are collected and memorised. Later on at the run-time of the model, for each test sample, called *query* in lazy learning, the model looks for samples that were memorised during the training phase in a neighbourhood of the query and builds a model based on them. The simplest example of such an algorithm is the k-Nearest Neighbours (kNN) method [48]. In the case of kNN, the model looks for the $k$ closest samples in the neighbourhood of the query and then makes a prediction based on these $k$ samples, which is a majority vote in the case of a classification problem or the mean value in the case of a regression task. Another possibility is to build a local predictive model in the neighbourhood of the query as is the case of Locally Weighted Learning discussed in Section 3.5.1.

In order to be able to build a local learning model, there are several additional aspects that have to be considered. A comprehensive study dealing with these aspects was presented in the context of lazy learning in [5]. These aspects include:

- The distance function: This function $d(\mathbf{x}, \mathbf{q})$, where $\mathbf{x}, \mathbf{q}$ stands for the training and query samples respectively, is applied to obtain the distances value between the samples. The most common distance function is the *Euclidean distance*:

$$d(\mathbf{x}, \mathbf{q}) = \frac{1}{m} * \sqrt{\sum_{j=1}^{m} (x_j - q_j)^2}. \tag{3.46}$$

  There are also different strategies for the application of the distance function possible. On one hand it can be a global distance function, which is the same for all samples, or on the

other hand the distance measure can differ for different queries, e.g. depending on the data density in the query neighbourhood.

- The kernel function: Given the distances between the samples, the kernel function describes the form of the local neighbourhood. As such, despite the same name, it plays a different role as the SVM kernel function described in Section 3.3.3. The size of the kernel, i.e. its extent in the sample space, is usually directly or indirectly a parameter of the function having important influence on the locality of the model. This parameter can either be a fixed value (e.g. as it is the case with RBFN) or change with the density of the data samples (e.g. in the case of kNN). A popular kernel form is the family of the exponential functions (e.g. Gaussian kernel).

- The local model: The local model, which is trained using the training data points from the neighbourhood defined by the kernel function, can be any kind of predictive method. The choice of the predictive technique can vary from simple average building (as in the case of kNN) to non-linear predictions using MLPs. However, since the complexity of the local data is usually quite limited, linear regression techniques are a popular choice for local models.

Theoretical justification of local learning approaches was given in [15]. In the cited work there is the *locality - capacity* trade-off introduced where the terms *locality* and *capacity* refer to the size of the neighbourhood, which is considered for training the local model, and to the complexity of the learning system (e.g. number of hidden units, pre-processing parameters, generalisation parameter), respectively. The stated trade-off is based on the balance between the capacity of a (global) algorithm and the number of training instances that defines the generalisation performance of the model [173]. The advantage of local learning is that the capacity of the model can be defined and controlled locally through the change of locality, i.e. the number of training samples.

An interesting discussion dealing with the link between the bias-variance dilemma (see Section 3.2) and local learning was published in [156]. In terms of local learning the problem of selecting an appropriate number of free parameters, e.g. number of hidden units, of a learning algorithm in order to find an optimum trade-off between bias and variance is transformed into the problem of the selection of the area of validity of a learning algorithm with fixed structure. In other words this means that instead of adapting the learning method one has to adapt the complexity of the region from which data for the training of the method is drawn and thus transforming the complex global optimisation into a simpler local one [156].

Besides the theoretical justification, local learning can also be found as an effective concept in biological systems. The evidence of locally focused information processing was found, for example, by identifying brain cells in the macaque vision system, which are responsible for the recognition of simple patterns in localised areas of the macaque's retina in [81]. These areas are called Receptive Fields. The term *Receptive Field* is adopted to refer to local partitions of the data space that, in the sense of the kernel function, build a connected neighbourhood.

From the process industry viewpoint, local learning is a promising strategy because often one can recognize different clusters in the space of the process data. These clusters correspond to different operating states of the process and it may be of benefit to train a local model for each of these states.

### 3.5.1 Local learning algorithms

This section briefly reviews a few local learning techniques that are relevant in the context of this work.

### Locally Weighted Learning

Locally Weighted Learning (LWL) [57] is an algorithm from the *lazy learning* family and as such it trains a new model for each test instance. The model is trained using training instances, from the local neighbourhood of the test samples. The instances closer to the test instance are allocated higher weights than more distant ones. The type and size of the kernel are important parameters since they define how *large* the local neighbourhood is. The drawbacks of the LWL technique are similar to the other lazy learning methods, it is mainly (i) the high computational demand at the prediction time since a new model must be built for each test instance; and (ii) the selection of the additional parameters like the kernel type, size, etc., which have a large impact on the performance of the model.

### Radial Basis Function Networks

Radial Basis Function Networks (RBFN) were already discussed in Section 3.3.2. From the point of view of local learning, the hidden units of the RBFN can be represented as modelling the Gaussian receptive fields in the input space. The output layer represents a linear model which builds the final prediction in dependency of the activation of the receptive fields.

### Modular neural networks

Jacobs et al. presented an interesting approach to supervised local learning [84]. The learning system consists of two different types of ANNs. The first type is trained using a subset of the available training samples. The authors call these models *local experts*. This term is now often used in terms of local learning to describe models that have been trained on subsets of the training data and will thus be adopted in this work as well. The second incorporated network type is a gating network, which weighs the outputs of the particular local experts. The gating network is a competitive multi-layer network with the number of hidden units equal to the number of local experts.

The training of the networks is done using the gradient descent technique. The global error function that is minimized is of the following form:

$$E = -log \sum_k p_k e^{-\frac{1}{2}\|y - y_k^{pred}\|^2}, \tag{3.47}$$

where $p_k$ are the weights predicted by the gating ANN, $y$ is the target value and $y_k^{pred}$ is the prediction of the $i$-th local expert given a test sample. This error function enforces the specialisation of the local experts and relaxes the couplings between the particular experts since every local expert is forced to predict the target value rather than a residual value as is the case with similar models.

### Locally Weighted Projection Regression

Locally Weighted Projection Regression (LWPR) [177] is a receptive fields-based local learning method for non-linear function approximation. Since LWPR stores all the necessary information needed for its incremental operation within the receptive fields descriptions, there is no need to store past data samples. The LWPR algorithm was derived from the Receptive Fields Weighted Regression (RFWR) [156] with particular focus on local dimensionality reduction and the operation (i.e. local experts building) in the locally reduced spaces.

Internally, a trained LWPR model consists of a set of receptive fields descriptions and locally valid PLS models. The receptive fields are Gaussian functions defined in a locally reduced space. Provided a new test (or query) sample, each of the local models makes a prediction $y_i^{pred}$ and activates the receptive fields. The activation of the receptive fields $w_i$ is applied as a weighting factor for the calculation of the final prediction $y^{pred}$:

$$y^{pred} = \frac{\sum_k w_k y_k^{pred}}{\sum_k w_k}. \tag{3.48}$$

Since LWPR is an incremental algorithm it uses the query samples not only to provide the prediction but also to update its structure, assuming the correct target value for the query sample is available. Updating the model structure consists of two tasks, namely updating the receptive fields, i.e. the Gaussian functions describing them, and the adaptation of the local regression models. The shape and size of the receptive fields is adjusted using the stochastic gradient descent approach in a penalized leave-one-out cross-validation cost function (see [156] for details). The incremental updates of the local regression models are achieved using a recursive least squares method. The authors use a modified version of the PLS algorithm that updates the regression models using a Newton-like method requiring the availability of simple statistics of the data only (see [177] for details).

One of the advantages of the LWPR algorithm is that the learning is completely localised, which means that the local models and receptive fields can be updated independently of each other. This fact dramatically simplifies the adding and pruning of the receptive fields.

## 3.6 Meta-learning

Meta-learning is a young research area, the first publications stating the term *meta-learning* in the context of machine learning were published in the early 1990s by J. Schmidhuber [160]. Due to its relatively recent introduction, the *meta-learning* term has not got a single unified definition yet and it is perceived in different ways by different researchers. Examples of definitions of the meta-learning term in machine learning are:

- Learning to learn [160]
- Studying how a learning system can increase efficiency through experience [178]
- Study of methodologies aimed to identify dynamical forms of biases [24]
- Process of exploiting of knowledge about learning [69]
- Accumulating experience on the performance of multiple applications of a learning system [179]
- Finding functions that map data sets to predicted data mining performance [138]
- Automatic process of acquiring knowledge that relates the empirical performance of learning algorithms to the features of the learning problems [143]

In summary, most of the previous definitions describe meta-learning as systems that learn how to improve the performance of a learning system by studying its past performance. Some of the approaches focus on overcoming the issue of a fixed bias (see below for the definition of bias in

the meta-learning context) by introducing techniques for dynamic bias selection. The practical approaches to achieve the meta-learning goals vary significantly but before reviewing the most popular of them some theoretical aspects, which will allow the explanation of the approaches, need to be introduced.

### 3.6.1 Meta-learning theory

The theoretical considerations presented in this work are based on the review of meta-learning published in [178] and [24]. For the purpose of the introduction to meta-learning, the general theoretical concept from Section 3.2 needs to be extended.

The learning process in meta-learning is often described as mapping from the data space $\mathcal{S}^{train}$ into a hypothesis space $\mathcal{H}_L$ of the learning algorithm $L$ in the following way:

$$L : \mathcal{S}^{train} \rightarrow \mathcal{H}_L. \tag{3.49}$$

The outcome of the learning is a hypothesis $h \in \mathcal{H}_L$. The meta-learning community also uses the term *bias*, however the notion of this term is different to the one introduced earlier in Section 3.2. In meta-learning, the *bias* of a learning algorithm refers to a set of assumptions that has an influence on the hypothesis space $\mathcal{H}_L$ of the learning algorithm. This can be, for example, the learning algorithm itself, the parameters of the algorithm (e.g. number of hidden units of a multi-layer perceptron), initialisation of the algorithm (e.g. initial weights of the multi-layer perceptron), data pre-processing and many others. The bias is a sum of all these effects and it defines what can and what cannot be learnt by $L$. Lets assume, for example, the case when the correct target concept $\phi$ lies outside of the hypothesis space $\mathcal{H}_L$ because the bias of the learning algorithm is too *strong*. In this case, the learning process cannot lead to successful learning of the target concept. This can happen when a linear learning algorithm (e.g. linear regression) tries to learn a non-linear mapping. On the other hand, in a overly large hypothesis space, it might be difficult to find the correct hypothesis $h$ given the limited size of the training data set $\mathcal{S}^{train}$. Incorrect bias selection can also lead to poor generalisation performance, i.e. data overfitting or overgeneralisation [24, 8]. These facts demonstrate the crucial role of the selection of the correct bias, where correct bias means that the target function lies within the hypothesis space $\mathcal{H}_L$ of the learning algorithm $L$.

The traditional learning scenario that was considered so far, i.e. selection of an optimal learning algorithm, data pre-processing and training the learning algorithm, is in meta-learning referred to as *base learning*. There are many problems with such an approach and these can be mostly related to the fact that the learning algorithm that is dealt with has a fixed bias. The bias can be manipulated by pre-processing the data (e.g. by performing feature selection) or by searching for parameters of the learning algorithm, which moves the hypothesis space closer to the target function. Furthermore, an implication of the *no-free-lunch* theorem is that there is no bias that leads to favourable performance when averaged over all possible learning tasks [68]. This means that there is no universal bias for every learning task and the bias has to be selected for each learning task separately.

Dealing with this problem is the goal of meta-learning. Meta-learning systems are dynamically trying to find the optimal bias that is necessary to solve the given task. This is achieved by collecting *meta knowledge* and providing mechanisms that exploit this knowledge for dynamic bias selection. Some approaches to achieve this goal are described in the following sections.

### 3.6.2 Meta-learning approaches

#### Meta knowledge acquisition approaches

The most straight-forward way of collecting meta knowledge is to extract features describing the underlying tasks and to link these with the performance of different learning algorithms using a meta-learner.

Obviously, there are several possibilities for the extraction of the features describing the tasks. The most popular way is to use general (e.g. number of samples, number of features), statistical (e.g. mean value, variance, kurtosis of the features) and information theoretic (e.g. class entropy) characteristics of the data sets defining the underlying task. There are several publications dealing with the extraction and evaluation of this kind of meta knowledge [16, 119, 99, 23]. Once the meta features describing the tasks are extracted, experiments with several learning algorithms, which act as base learners, have to be performed (see [98] for a large study dealing with algorithmic performance). The outcomes of the experiments are the relative performances of the learning algorithms on different tasks. This data is used for building the output space for the meta-learning task. Having this data, a learner, which links the meta features with the performance of the base learners, can be trained. It is obvious that the meta learner is again biased and the same issue that was dealt with at the base level is present. This fact highlights one of the conceptual issues of meta-learning because in order to deal with the bias selection at the meta-level there is a meta-meta-level learner required and so on. Nevertheless, it has been shown in several cases that meta-learning can be beneficial for improving the generalisation performance. Two particular projects that focused on the descriptions of data sets in terms of statistical and information theoretic features are STATLOG [101] and METAL [99].

Another approach is to base the meta attributes on characteristics like the tree depth and tree width of decision trees that are trained to solve the task. These meta features are again used as an input for the meta learner, which links them to the performance of the base-learners [11, 10, 138].

Yet another approach to meta attribute building is called *landmarking* [141]. Landmarkers are simple learners whose performance is used to describe the underlying task. The aim of the meta learner is to link the performance of the landmarkers to the performance of the actual base-learners. Then, when confronted with a new task, the meta learner selects a base learner depending on the landmarker performances.

In [179], there is a *meta-learning architecture* presented that is based on the above approach. The architecture can be operated in two modes, namely in the *knowledge acquisition* and *advisory* modes. In the first mode the model collects knowledge about presented tasks and extracts the meta knowledge and the performance of the implemented base models. The gathered knowledge is exploited during the advisory mode, during which the model deals with new tasks (i.e. data sets). By checking the knowledge database, the model should be able to make recommendations for data pre-processing and model selection for the novel data set.

#### Learning from base learners

An example of a system where the meta learner learns from base learners is *stacked generalisation* [199]. In the case of such a system, there is a base learner trained for each data set. On the meta-level, the predictions of the base learners are connected to the original data variables and a new learner, i.e. meta-learner, is trained and its prediction is considered as the output of the system. Although most of the approaches focus on such two-layer hierarchy, a multi-level hierarchy can easily be built by training a set of meta-learners and by propagating their predictions to the next

level, i.e. meta-meta-level.

*Boosting* [49, 24] can also be considered as a meta-learning system. In this case, a set of individual learners is built by producing variations of the training data by sampling from the original training data under a probability distribution that is defined by the past prediction performance on the particular data samples, i.e. the higher error the more likely it is that the sample is drawn for the building of the next base learner. The final prediction is obtained by building a weighted combination of the predictions of the base learners, whereas the predictors with higher performance also get higher weights. This method can be considered as a meta-learning approach because it moves the hypothesis space of the newly built base learners to spaces where the previous base learners performed poorly [179].

### Other meta-learning approaches

There is a large number of other approaches to meta-learning which are in one or another way trying to find a beneficial hypothesis space $\mathcal{H}_L$ as it is done in the case of *learning to learn* [9], *dynamic bias selection* [196], and *bias learning* [8].

### Learnt topology gating artificial neural network

A particularly relevant meta-learning algorithm that was developed during this project is the Learnt Topology Gating Artificial Network (LTGANN) [93]. It is an extension of another work studying gating ANNs, which was also developed and published during this project [92]. The main goal of both works was to experiment with ensemble methods and meta-learning in the context of artificial neural networks. Both of the approaches are motivated by modular neural networks (see Section 3.5.1).

The earlier gating ANN deals with model selection, which in terms of ANNs means mainly the selection of hidden units, by training and combining networks with a random number of hidden units. In order to combine the set of the trained base models, there is another ANN trained, called gating ANN. The purpose of the gating ANNs is to estimate the prediction error of the base ANN on a sample-by-sample basis. The structure of the network is shown in Figure 3.3. The gating

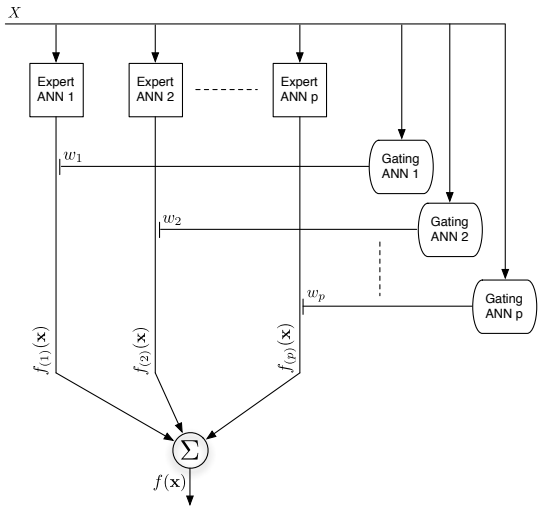



Figure 3.3: Gating Artificial Neural Network structure

networks were trained by using the following data samples:

$$\mathcal{S}^{gate,train} = \left\{(\mathbf{x}_i^{val}, w_i)\right\}_{i=1}^{n^{val}}, \tag{3.50}$$

$$w_{k,i} = \frac{1}{1 + \left(f_{(k)}(\mathbf{x}_i) - y_i\right)^2}, \tag{3.51}$$

where $n^{val}$ is the size of the base learner's validation data set, $\mathbf{x}_i^{val}$ is the $i$-th validation input sample, $f_{(k)}(\mathbf{x})$ the value predicted by the $k$-th predictor, $y$ the correct target value and $w_i$ a weight based on the inverse prediction error. Training the gating ANNs with this data corresponds to building meta-learners that should be able to predict the performance of the base learners given the input data. The final prediction of the model is a weighted sum of the particular predictions with weights predicted by the gating ANNs:

$$f(\mathbf{x}_i) = \sum_{k=1}^{p} w_{k,i} f_{(k)}(\mathbf{x}_i). \tag{3.52}$$

The results presented in [92] were obtained by evaluating the model on the industrial drier data set (see Appendix B.1). The analysis has shown several advantages of using the gating networks. While adding new base learners (add gating networks) to the model, the prediction error converged faster than the prediction error achieved by making mean combination of the base learners. The convergence level of the proposed model was also shown to be significantly lower than the one of the mean combination of the base learners.

The LTGANN extends the previous model by adding the ability to learn the relation between the performance and the different ANN topologies, i.e. numbers of hidden units. This knowledge is of advantage since it can be exploited when adding a new base learners to the model. Initially, the topology of the base learners (BANN) and meta-learners, i.e. gating ANN (GANN), is determined randomly by drawing the numbers of hidden units from a uniform distribution $\mathcal{U}(H_{BANN})$ and $\mathcal{U}(H_{GANN})$, where $H_x$ is a pre-defined range of possible hidden number units for the base and gating networks respectively. After generating, training and evaluating the performance of the networks with the number of hidden units $h_{BANN} \in H_{BANN}$ and $h_{GANN} \in H_{GANN}$, the relative performances $q_{BANN}$ and $q_{GANN}$ are used to modify the originally equal distribution towards the conditional distributions for both topologies $P(H_{BANN}|q_{BANN})$ and $P(H_{GANN}|q_{GANN})$:

$$P(H) \xrightarrow{init.} \mathcal{U}(H) \xrightarrow{learning} P(H|q). \tag{3.53}$$

With each new step (i.e. adding new base-gating network pair), the up-to-date distributions are used to generate the topologies of the new networks.

The learning of the conditional performance distributions can be interpreted as a meta-learning approach as the technique automatically learns the bias (i.e. the topology) of the ANN. In the case of LTGANN, meta-learning takes place at three different levels in the form of: (i) learning the conditional performance distributions of the base ANN; (ii) learning the same for the gating ANN; and (iii) using the gating networks to model the area of expertise of the base networks.

Applying the topology learning led to performance improvement of the model, which was shown in experimental evaluation based on two industrial data sets [91]. Comparing the two approaches discussed in this section (compare [92] and [91]), shows that by learning the topologies one can achieve faster convergence, i.e. lower number of networks need to be trained, while additionally improving the converged performance level.

## 3.7 Concept drift and adaptivity

Concept drift detection and handling is a field of machine learning focusing on dealing with changing environments. Concept drift was described as: "The change of the target concept, which is caused by changes in some hidden context." [197]. In terms of process industry data, the hidden context can be, for example, the environmental conditions, e.g. seasonal changes, the quality of the input materials or in other words anything that has an influence on the process and thus on the data. One can distinguish between two types of concept drift, namely *virtual* and *real* concept drift [180]. In case of virtual concept drift, there is a change in the distribution of the data only while the concept itself remains stable. In the terminology introduced in Section 3.2 this means, that input and/or output space change but the target function $\psi$ remains constant:

$$\mathcal{X} \rightarrow \mathcal{X}' \text{ and } \mathcal{Y} \rightarrow \mathcal{Y}' \text{ and } \phi = \mathcal{X}' \rightarrow \mathcal{Y}'. \tag{3.54}$$

Whereas in case of real concept drift there is an actual change of the concept, i.e. relation between the input and target data. Both of the discussed concept drift types require an adaptation of the model in order to keep an acceptable level of performance after the drift. In this case the target function $\phi$ mapping the input space on the output space is time variant:

$$\phi(t) = \mathcal{X} \rightarrow \mathcal{Y}. \tag{3.55}$$

In general, dealing with concept drift consists of two tasks: (i) *concept drift detection* [64], which can be done by the identification of the symptoms such as poor model performance, etc.; and (ii) *concept drift handling*, which in most cases requires model adaptation. According to [169], there are three different approaches to adaptation in order to handle concept drifts. These are:

- instance selection [124, 12]
- instance weighting [104, 64]
- ensemble methods [161, 189]

The following sections provide some examples of different approaches to concept drift handling while paying particular attention to techniques that have been applied to adaptive soft sensing.

#### Instance selection

The most commonly used approach falling into this category is the moving window technique. In this case, the model is updated or retrained using a set of the most recent data samples. The drawback of the method is that several critical parameters, like the size of the window and the interval between the updates, have to be set in order to achieve good adaptation performance. The most significant issue is that using fixed window size, which is often the case, avoids the concept drift detection task as the window of fixed size is moving at constant speed and there is no guarantee that the samples within the window actually correspond to the current data concept. This can be solved by applying an adaptive window size technique. Such an approach was studied in [113], where windows of three different sizes compete with each other and the one providing the best performance improvement is selected. Another work proposing a dynamic window size is based on an entropy indicator used for the estimation of the optimal window size [180].

In practical circumstances, the window size is usually estimated by the model developer in an ad-hoc manner. The danger of such an approach is that by selecting too short window, the

model may adapt to noise. On the other hand, with too long window covering several concepts, the model may also fail to adapt properly. An example of a work where the discussed issue was approached is [105], where the window size was adapted in order to minimize the estimated error of the classifiers on new arriving samples.

Examples of adaptive soft sensors based on the moving window technique can be found in [188] where a process monitoring soft sensor based on the Fast Moving window PCA is presented or in [206] where a batch process monitoring soft sensor based on the Moving window PCA have been published.

### Instance weighting

An alternative approach to adaptive modelling is using temporal weighting of samples [104, 64]. In this case, there is a weight assigned to each training sample. In order to keep the model up-to-date, recent data samples get higher weights than older ones. This method has similar issues with parameter estimation as the moving window technique. Focusing on the previous example, where the weights are assigned according to the age of the sample, the critical parameter is the speed of the temporal decay of the sample weights. Similar to the moving window technique, it also does not provide any mechanism for concept drift detection unless the decay rate itself is adapted according to the speed of concept change. Another drawback of this approach is that it cannot be combined in a straightforward way with the modelling techniques without their modifications. Often there is also a need for the implementation of a special training algorithm that takes the weights into account. Nevertheless, there are several modifications of PCA/PLS algorithms providing adaptation mechanisms based on weighted samples. Examples of such techniques and soft sensors based on them are: (i) Recursive PLS [145]; (ii) Recursive Exponentially Weighted PLS [37]; and (iii) Recursive PCA [117]. In [30], there is also a technique for dynamic adjustment of the forgetting factor presented.

### Ensemble methods

The last approach for adaptive model building discussed here is based on the ensembles concept. A particular benefit in the case of concept drift handling using ensemble methods is that the adaptation can be shifted from the adaptation of the ensemble members to the adaptation of the combination weights. Concept drift handling using ensemble methods can have different forms. One can, for example, simply change the combination weights according to the drifting data. The drawback of this approach is that it assumes that there is at least one member of the ensemble that can make an appropriate prediction. If this cannot be guaranteed, there is an additional requirement to add new members to the ensemble that are trained on the latest available data. Such an approach was published in [184], where a set of classifier is trained and combined using a weighted sum approach. The combination weights are related to the estimated predictions performance of the ensemble members. As the work deals with streaming data, the estimated performance is updated with the new incoming samples and correct target values such that a new classifier can be trained during the run time of the model. This results in the adaptation of the whole system by changing the combination weights and updating the set of classifiers.

In [32], there was similar system with different combination weights adaptation techniques presented. In this case the weights are calculated and updated to maximise the likelihood of the data and model parameters under the Bernoulli distribution. The Bernoulli distribution is used because the targeted problem is a binary classification task. The system was shown to provide better performance in comparison to the previously mentioned one. However, a practical imple-

mentation may suffer from a similar issues as the instance selection techniques since the authors train a fixed number of classifier on chunks of data of fixed size.

As for adaptive soft sensors, apart from the works published during this project [91, 94] there are no other soft sensors reported that are based on concept drift handling using ensemble methods.

## 3.8 Summary

The purpose of this chapter is, on one hand, to embed the soft sensor development into the machine learning theory and, on the other hand, to discuss some promising machine learning concepts that can be applied in order to improve the performance of soft sensors as well as to extend their lifetime.

The considered concepts are the ensemble methods, local learning, meta-learning and concept drift.

In the case of ensemble methods, there is a set of models trained and combined to form the final prediction. One of the major findings of the ensemble methods research is that in order to build a successful ensemble, there is a certain degree of diversity required among the ensemble members.

One of the ways for creating diversity is by applying local learning, i.e. to train a set of local models, an ensemble, each of which is focused on a limited part of the data space. In order to combine the predictions of the local models, a combiner has to estimate the performance of particular local models and combine their prediction according to the estimated performance. An adaptive soft sensor based on such an approach is presented in the next chapter.

The next part of this chapter dealt with meta-learning. For the purpose of this work, any technique that operates at a level higher than the predictive methods will be understood as meta-learning. As such, the ensemble-based local learning techniques are also part of meta-learning.

Concept drift detection and handling provides several ideas for the building of adaptive models. Interestingly, one type of approach is closely related to ensemble methods. This method is particularly efficient because there is no need to adapt the predictive models themselves as the adaptation happens at the level of the combiner.

An interesting observation from this chapter is that one can find ensemble methods across all of the discussed concepts. Ensemble methods can be used to deal with local learning, they are one of the meta-learning approaches and can also be applied to deal with concept drift. This fact indicates the significant role that this technique is going to play for the development of robust and adaptive soft sensing algorithms in this work. A first attempt for such an algorithm is presented in the next chapter.

# Chapter 4

# Simple adaptive soft sensing algorithm

## 4.1 Introduction

This chapter is the first practical step towards the development of the concept for robust and adaptive soft sensors by proposing a local learning-based algorithm for on-line prediction soft sensing. Local learning is investigated because it is theoretically a promising method since the process data often consist of separate partitions as a result of different operating states of the underlying processes. By identifying the distinct states of the data, one can build simple models that only focus on the modelling of parts of the data. Consequently, during the on-line phase the challenging task is the identification of the most appropriate model(s) and taking only their predictions into account.

In this chapter, such a technique is also shown to be providing an elegant possibility for the implementation of the model adaptation. However, before introducing the algorithm, a modified version of the soft sensor development methodology, which incorporates the adaptive behaviour of the soft sensor, is proposed.

The structure and dependencies between the sections of this chapter are shown in Figure 4.1.

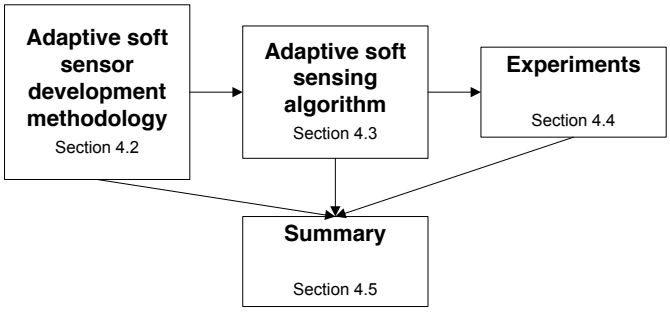



Figure 4.1: The structure of this chapter

## 4.2 Methodology for adaptive soft sensor development

In order to be able to deal with the requirements of the adaptive soft sensor development, the general methodology discussed in Section 2.4.2 has to be slightly modified. Another purpose of the modifications is to change the methodology in order to support automated soft sensor development as required by the aims of this project.

The updated methodology assumes that a large part of the soft sensor development, especially the data pre-processing and model validation, selection and maintenance, are automated and there is only minimal need for manual intervention. However, realistically there will be some need for manual intervention, which is also taken into account by this methodology. For this purpose the model life cycle is split into the following three steps (see Figure 4.2 and 4.3): (i) Manual pre-processing phase; (ii) Model building phase; and (iii) Real-time operation. These are individually discussed in the following paragraphs.

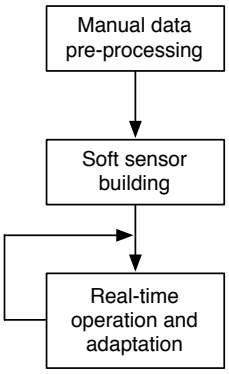



Figure 4.2: The life-cycle of an adaptive soft sensor

**Manual data pre-processing:** This step roughly corresponds to the first two steps of the methodology in Section 2.4.2. The aim is to cover the part of the model development that still requires some manual data inspection and pre-processing. Typically, this will include: (i) First data inspection; (ii) Selection of historical data; and (iii) Steady state detection.

The input to this phase is the raw process data as they are read from the Process Information Management System (PIMS). In this work we assume that the data are collected and provided automatically by the PIMS in batch mode.

The outcome of this stage is a batch of historical data that is further used for the training and validation of the soft sensor.

**Soft sensor building:** This phase receives the manually pre-processed historical data as input. The task of this phase is to provide the trained model for the next phase. During this phase all the steps should be automated as far as possible and the manual intervention of the model builder should be minimised. The particular steps within this phase are: (i) data pre-processing; (ii) model selection; (iii) model training and validation, which are covered by the third and fourth step in the original methodology (refer to Figure 2.4).

**Real-time operation and adaptation:** The trained soft sensor has to provide its predictions as a response to the incoming data stream in this phase. Additionally, in a scenario with (occasionally) available correct target values, the soft sensor should be able to assess its performance. In the case when a deterioration of the performance level is recognised, the model should take appropriate measures to maintain a stable performance level, i.e. adapt itself to the changing environment.

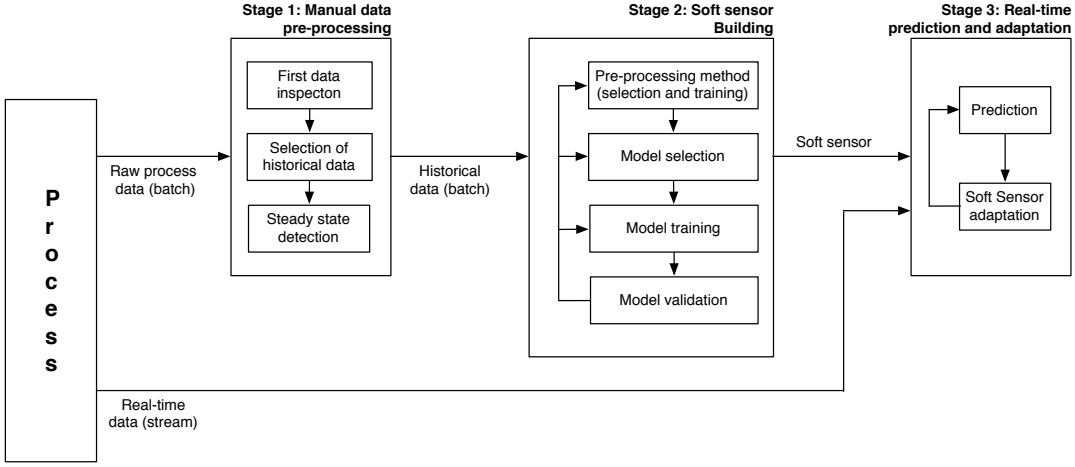



Figure 4.3: Detailed overview of the proposed soft sensor development and maintenance methodology

## 4.3 Adaptive soft sensor for on-line prediction

This section presents a soft sensor algorithm developed following the methodology discussed in the previous section. The soft sensor development, operation and maintenance can therefore be split into three different steps.

The first, i.e. the manual data pre-processing, is kept minimal and consists of removing some obviously corrupt variables, etc. The details for each of the treated data sets can be found in Appendix B. The second part makes use of the fully labelled historical data. This data is used for the initial training and performance evaluation of the soft sensor. After finishing the training, the soft sensor uses the real-time input data stream to make on-line predictions of the target values during the third phase. When available, the measured target values can be used for the adaptation of the soft sensor during the on-line phase.

The proposed soft sensor development and operation can be split into the following steps:

- Construction of receptive fields
- Training of local models/experts
- Building of receptive field descriptors
- Combination of local experts
- On-line operation and adaptation of the model.

### 4.3.1 Receptive fields construction

The aim of the receptive field building is to divide the historical data into partitions representing different data concepts (see Section 3.7 for the *concept* definition). The notion of a concept is linked to the area where a model, called *landmarker*, provides *constant* performance. The decrease of performance of the landmarker is interpreted as a new data concept that triggers the building of a new receptive field.

Provided the historical data set $\mathcal{D}^{hist}$, the first step of the algorithm is training the landmarker using samples from an initial window $\mathcal{D}^{init}$, which is a subset of the historical data:

$$\mathcal{D}^{init} = \{X, \mathbf{y}\} = \{(\mathbf{x}_i, y_i)\}_{i=a}^{a+n^{init}}, \tag{4.1}$$

where $a$ is the index of the first sample in the current receptive field and $n^{init}$ is the length of the initial window (an input parameter of the algorithm).

Provided the initial set, the landmarker $f^{lm}$ can be trained and the residual vector $\mathbf{r}^{init}$ of the landmarker's prediction on the training data can be calculated:

$$\mathbf{r}^{init} = \mathbf{y}^{init} - f^{lm}(X^{init}), \tag{4.2}$$

where $f^{lm}(X^{init})$ are the predictions of the landmarker.

The next step is shifting the window one step forward ($s = 1$), while keeping its size constant:

$$\mathcal{D}^{shifted} = (X^{shifted}, \mathbf{y}^{shifted}) := \{(\mathbf{x}_i, y_i)\}_{i=k+s}^{k+s+n^{init}} \quad \text{with } s = 1, \tag{4.3}$$

and calculating the new residual values $\mathbf{r}^{shifted}(s)$ of the landmarker's prediction using the shifted data window:

$$\mathbf{r}^{shifted}(s) = \mathbf{y}^{shifted} - f^{lm}(X^{shifted}) \tag{4.4}$$

Following this, the two residual vectors ($\mathbf{r}^{init}$ and $\mathbf{r}^{shifted}(s)$) are tested for a statistically significant difference using the t-test [73]. This test was chosen because the residuals can be, ideally, assumed as normally distributed. The t-test is looking for a significant difference in the mean values of the two residual vectors and so it is able to identify a significant change in the performance of the landmarker as an effect of the concept drift. As long as the null hypothesis remains valid, it can be assumed that the performance of the landmarker on the data within the shifted window is comparable to the performance on the training data and thus that the data samples, within the shifted window $\mathcal{D}^{shifted}$, belong to the same concept as the samples from the initial window $\mathcal{D}^{init}$. This procedure is repeated, i.e the window is shifted, as long as the null hypothesis of the significance test remains valid:

$$s_i^{final} = \underset{s \in [1,\ldots,n^{train}-k]}{\operatorname{argmin}} (ttest(\mathbf{r}^{init}, \mathbf{r}^{shifted}(s)) == 1), \tag{4.5}$$

where $n^{train}$ is the number of samples in the historical data set and $s^{final}$ corresponds to the first sample for which the t-test rejects the null hypothesis and thus there is a significant difference in the residuals. The significance level of the t-test is an important parameter because it has a strong influence on the size of the receptive fields. However, in the experiments reported in this chapter the significance level is fixed to $0.05$ because the same effect can be achieved by manipulating the size of the initial window size $n^{init}$, which is already an input parameter of the algorithm.

Finally, the receptive field is built by the following data samples:

$$\mathcal{D}_r^{RF} = \{(\mathbf{x}_i, y_i)\}_{i=k}^{k+n^{init}+s_i^{final}-1} \tag{4.6}$$

and the algorithm can move to the next receptive field by constructing a new initial window. This is constructed by taking the last $n^{init}$ samples of the previous receptive field (i.e. using the last shifted window of the previous receptive field) which results in a partial overlap of the receptive

fields:

$$\mathcal{D}^{init} := \mathcal{D}^{shifted} = \{(\mathbf{x}_i, y_i)\}_{i=a+s^{final}}^{a+s^{final}+n^{init}}, \tag{4.7}$$

$$a = a + s^{final}. \tag{4.8}$$

The procedure of receptive field building is graphically illustrated in Figure 4.4.

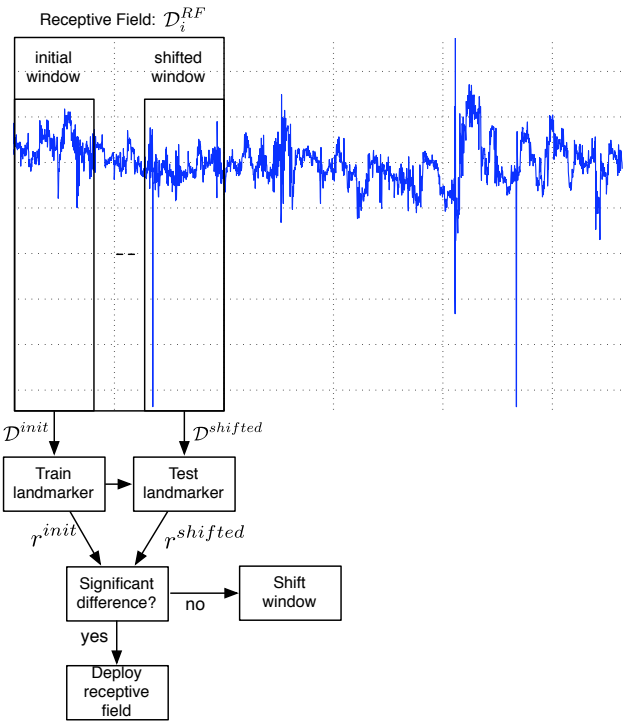



Figure 4.4: A novel receptive field construction process based on concept drift detection

The outcome of this stage is a set of receptive fields $\mathcal{D}^{RF}$, each corresponding to a concept of the historical data:

$$\mathcal{D}^{RF} := \left\{\mathcal{D}_i^{RF}\right\}_{i=1}^{r}, \tag{4.9}$$

where $r$ is the number of built receptive fields.

### 4.3.2 Local experts training

After identifying the receptive fields, one model, called *local expert* $f^{LE}$, is trained for each of the receptive fields. The modelling technique for the local experts is relatively unconstrained. Any computational learning technique can be applied for the local experts. However, as each of the receptive fields represents one concept of the historical data as it is seen from the perspective of the landmarker, a modelling technique, which is related to the landmarker should be selected, i.e. if the landmarker is a linear regression model, the local expert should also be a linear modelling technique.

It is beneficial to build the local experts in locally pre-processed data spaces. In particular the pre-processing using the PCA applied in the experiments in this chapter is very useful. It reduces the dimensionality of the input data spaces by removing co-linearity from the data, and thus reduces the number of local expert descriptors needing to be built.

After this step there is a set of trained local experts $\mathcal{F}^{LE}$:

$$\mathcal{F}^{LE} := \left\{ (f_{(k)}^{LE}) \right\}_{k=1}^{p}, \tag{4.10}$$

where $p$ is the number of local experts[1].

### 4.3.3 Local experts descriptor building

The next step toward the final model is building descriptors for the local experts, which will be later used to estimate the prediction accuracy of the local experts for the given input sample.

As such, the aim of the descriptors is to describe the area of expertise of the particular local expert. This is approached by building two-dimensional probability density functions $L_{k,j}$, where $j$ is the index of the input variable (after the PCA pre-processing) and $k$ the index of the local expert (or receptive field). The descriptors are constructed using a weighted two-dimensional Parzen window method [135]:

$$L_{k,j} = \frac{1}{\left\| \mathcal{D}_k^{RF} \right\|} \sum_{\mathbf{x}_i \in \mathcal{D}_k^{RF}} w_k(\mathbf{x}_i) \Phi(\mu, \Sigma) \tag{4.11}$$

where $\left\| \mathcal{D}_k^{RF} \right\|$ is the number of samples in the $k$-th receptive field, $w_k(\mathbf{x}_i)$ is the weight of the $i$-th sample point (for more details see below), $\Phi(\mu, \Sigma)$ is a two-dimensional Gaussian kernel function with mean value at the positions defined by $\mu := \{x_{i,j}, y_i\}$ and variance matrix $\Sigma$ (a diagonal $2 \times 2$ matrix with the kernel width $\sigma$ at the diagonal positions - an input parameter of the algorithm). The kernel width defines the size of the neighbourhood, which is influenced by each sample. For simplicity reasons, the variance is kept equal in both dimensions but the approach can be easily extended to a more general case with different kernel widths along the two dimensions. Figure 4.5 shows an example of the descriptor for three different settings of the kernel size $\sigma$. One can see that with smaller kernel sizes the descriptor gets more detailed, which involves a potential danger of low generalisation capability and overfitting of the descriptor. On the other hand, two large kernels can lead to too strong generalisation and loss of detail. The peaks indicate the area of the input-output space where the given local expert performs better than the remaining local experts.

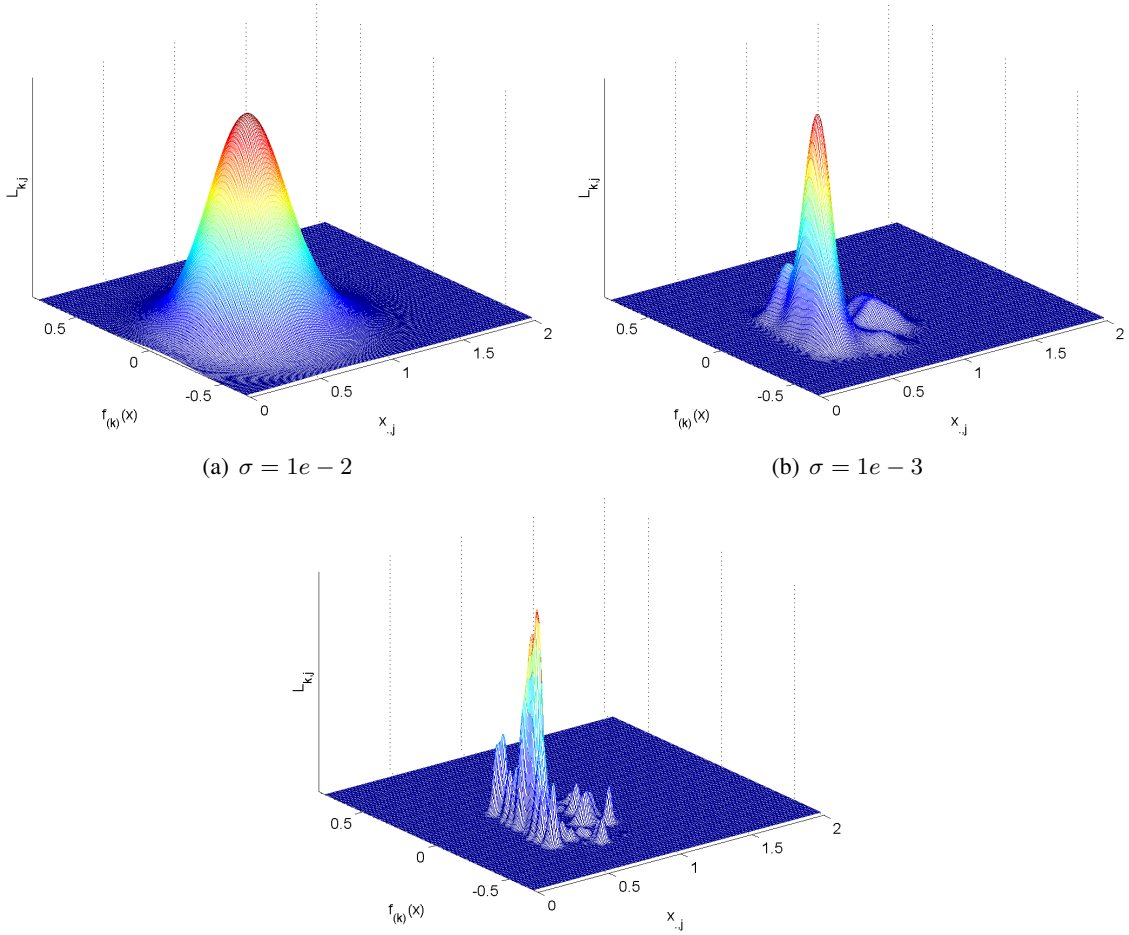
At this position it should also be noted that in the further implementation, the descriptors are built in input-output spaces, which are the outcome of local PCA pre-processing. The benefits of this approach are that the descriptors are built in lower dimensional spaces. Since the PCA is also trained locally for each receptive field, it assures that it extracts locally relevant information and builds a sub-space that describes the receptive field better than global pre-processing would allow.

The weights $w_k$ for the construction of the descriptors (see Equation 4.11) are set proportionally to the inverted quadratic prediction error of the local experts:

$$w_k(\mathbf{x}_i) = \exp(-(f_{(k)}^{LE}(\mathbf{x}_i) - y_i)^2) \tag{4.12}$$

Weighting the contribution of each sample by the prediction performance of the corresponding local experts assures that the descriptors model the local experts' area of expertise in the input-output space and as such can be later sampled to estimate the local experts' performance given the input data and its prediction.

[1] For this implementation $p = r$, because there is one local expert build for each receptive field.




(a) $\sigma = 1e-2$ (b) $\sigma = 1e-3$

(c) $\sigma = 1e-4$

Figure 4.5: Local expert descriptor $L_{k,j}$ with different settings of $\sigma$

The final descriptor $\mathcal{L}$ is a set of $l \times r$ two-dimensional descriptors, with $l$ being the number of input variables (after PCA pre-processing) and $r$ the number of receptive fields:

$$\mathcal{L} := \{(L_{k,j})\}_{k=1;\, j=1}^{r;\, l} . \tag{4.13}$$

### 4.3.4 Local experts combination

At this stage there is a set of trained local experts $\mathcal{F}^{LE}$ and receptive field descriptors $\mathcal{L}$ available. From the set of predictors $\mathcal{F}^{LE}$, each of its members $f^{LE}$ is making predictions of the target value given the input samples. In order to obtain the final predictions $y^{final}$ these predictions have to be combined.

In what follows, the combination method will be described in the Bayesian framework, in general terms the combined prediction is a weighted sum of the local experts' predictions:

$$y^{final} = \sum_{k=1}^{r} v_k(\mathbf{x}, f_{(k)}^{LE}(\mathbf{x})) f_{(k)}^{LE}(\mathbf{x}), \tag{4.14}$$

where $f_{(k)}^{LE}(\mathbf{x})$ are the predictions given the on-line data sample and $v_k$ the combination weight of the $k$-th local expert's prediction. These weights are read from the local expert descriptors $\mathcal{L}_k$. Since the descriptors store maps of the local experts' performance in the input-output space, they can be interpreted as an effective estimation of the prediction performance of the local experts for the given on-line data sample. The weights can be expressed as the posterior probability of the $k$-th local expert given the sample $\mathbf{x}$ and the prediction of the local expert for this sample $f_{(k)}^{LE}(\mathbf{x})$:

$$v_k(\mathbf{x}, f_{(k)}^{LE}) = p(k|\mathbf{x}, f_{(k)}^{LE}) = \frac{p(\mathbf{x}, f_{(k)}^{LE}|k)p(k)}{\sum_k p(\mathbf{x}, f_{(k)}^{LE}|k)}, \tag{4.15}$$

where $p(k)$ is the a priori probability of the $k$-th local expert (in our implementation equal for all local experts but in general it can be used to prioritise between them), $\sum_k p(\mathbf{x}, f_{(k)}^{LE}|k)$ is a normalisation factor and $p(\mathbf{x}, f_{(k)}^{LE}|k)$ the likelihood of $\mathbf{x}$ and the local expert, which can be calculated by reading the descriptor of the $k$-th local expert:

$$\begin{aligned} p(\mathbf{x}, f_{(k)}^{LE}|k) &= \prod_{j=1}^{l} p(x_{\cdot,j}, f_{(k)}^{LE}|k) \\ &= \prod_{j=1}^{l} L_{k,j}(x_{\cdot,j}, f_{(k)}^{LE}(\mathbf{x})). \end{aligned} \tag{4.16}$$

Equation 4.17 shows that the descriptors $L_{k,j}$ are read at positions that are given by the scalar value $x_{\cdot,j}$ of the $j$-th variable of the sample point $\mathbf{x}$ and at the position of the predicted output $f_{(k)}^{LE}(\mathbf{x})$ of the $k$-th local expert. Sampling the descriptors at the positions of the predicted outputs may be potentially ineffective because the predicted value does not necessarily need to be similar to the correct target value. However, as the correct target values are not available during the on-line phase at the time of the prediction, this is the only way to read the values from the descriptors. Furthermore, the rationale for this approach is that the local expert is likely to make the correct prediction if it generates a prediction that conforms with an area that was occupied by a large number of accurate predictions during the training phase.

### 4.3.5 Soft sensor adaptation

One of the benefits of the proposed approach is that it provides several possibilities for the implementation of adaptive mechanisms. At this stage a simple but powerful adaptation mechanism, which adapts the combination weights $v$ (see Equation (4.15)), is discussed. This goal is achieved by adapting the local expert descriptors $\mathcal{L}$. The proposed technique does not require any storage of past data samples and as such works fully incrementally on a sample-by-sample basis. However, because no new models are being deployed during the on-line phase, a limitation of this type of adaptation is that the model requires that the on-line data represents a data context that was present during the training phase and that there was a local expert trained for this context in the past. This fact is going to be further validated in Section 4.4.

#### The adaptation algorithm

As mentioned above, this adaptation technique adapts the descriptors of the receptive fields during the on-line phase. The descriptors are modified each time a correct target value $y$ is received.

Provided this value, a feedback about the performance of the local experts in the form of the quadratic error $e_{(k)}$ of the $k$-th local expert's prediction $f_{(k)}^{LE}$ is calculated:

$$e_k(\mathbf{x}) = (y - f_{(k)}^{LE}(\mathbf{x}))^2. \tag{4.17}$$

The error is further mapped on a performance index $u_k$:

$$u_k = \exp\left(-\frac{e_k - \text{med}(\mathbf{e})}{\text{med}(\mathbf{e})\log(2)}\right), \qquad \text{with } \mathbf{e} = [e_1, \ldots, e_p] \tag{4.18}$$

where $\text{med}(\mathbf{e})$ is the median squared error across the local experts and $p$ the number of local experts. The above mapping transforms the prediction error in such a way that the best performing local expert receives a weight equal to 1 and the weights of the remaining local experts decay exponentially with the increasing error, whereas the median error is mapped to the value $0.5$.

This mapping function, together with Equation 4.19, leads to a decrease of the neighbourhood of the current sample within the receptive field descriptors (see Figure 4.6(a) for the adaptation mask in this case) for local experts providing weak performance. Contrary to this, descriptors of local experts whose performance is better than the average are increased in the neighbourhood of the current sample $\mathbf{x}$ (see Figure 4.6(b) for an example of such an adaptation mask). This approach leads to an increase of areas within the descriptors where the local expert performs well and to a decrease of such areas that can be better predicted by another local expert.

The following equations describe the adaptation process:

$$\Delta L_{k,j} = \Phi(\mu, \Sigma)(u_k - 0.5), \quad \text{with } \mu = \left\{\mathbf{x}, f_k^{LE}(\mathbf{x})\right\}, \tag{4.19}$$

where $\Delta L_{k,j}$ is a two-dimensional Gaussian adaptation mask for the $j$-th variable and $k$-th receptive field (or local expert). Further on $\Sigma$ is the variance matrix for the Gaussian kernel. These values of the $2 \times 2$ matrix (consisting of $\sigma^{adapt}$ at the diagonal positions) define the size of the neighbourhood of the current sampling point that is being modified by the adaptation mask (an input parameter of the algorithm).

Finally, the descriptors can be adapted using the modification masks in the following way:

$$L_{k,j}^{new} = L_{k,j} \cdot \Delta L_{k,j}, \tag{4.20}$$

where $\cdot$ is the Hadamard matrix product.

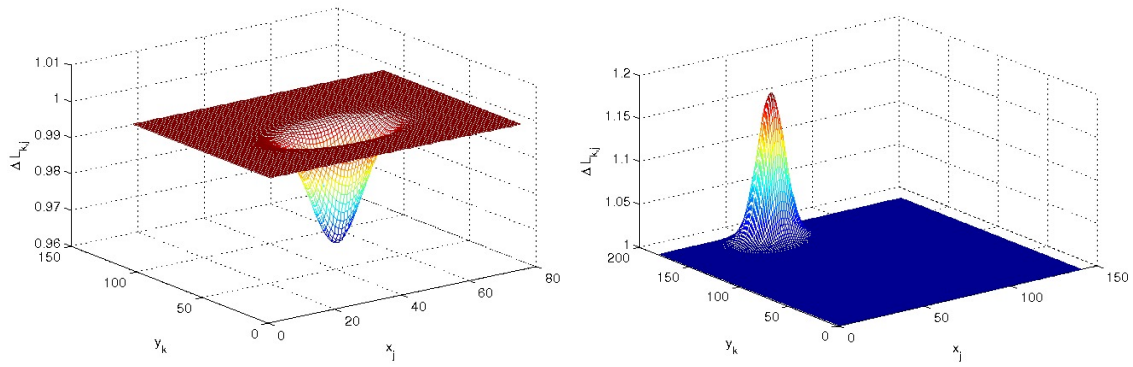
## 4.4 Experiments

In this section, there are several types of experiments dealing with industrial data sets provided. The analysed data sets are:

- Industrial drier, see Appendix B.1 for details
- Thermal oxidiser, see Appendix B.2 for details
- Catalyst activation, see Appendix B.3 for details

The predictive techniques that are analysed are:

- PCA+MLP: Multi-Layer Perceptron with Principal Components Analysis pre-processing is




(a) (b)

Figure 4.6: Adaptation masks $\Delta L_{k,j}$ for the modification of the local expert descriptors

a state-of-the-art approach to soft sensing and one of the most common soft sensors (see Section 2.4.3)

- LWPR: Locally Weighted Projection Regression is a PLS (another popular soft sensing method) and at the same time a local learning-based algorithm with an interesting adaptation method.

- LASSA: Local Adaptive Soft Sensing Algorithm is the method proposed in this chapter.

Each of the techniques is first evaluated in a non-adaptive environment. After that the adaptation techniques of the different techniques are tested. The LWPR and the LASSA techniques use the adaptation approaches discussed in [177] and Section 4.3.5 of this work respectively. In the case of the PCA+MLP soft sensors, the applied adaptation mechanism is the moving window technique.

The general purpose of the experiments is to evaluate the above algorithms in terms of the difficulties with parameter selection, parameter sensitivity and their adaptation capability in an environment with a very low amount of available expert knowledge. As such, the goal of the experiments is rather to study the characteristics of the soft sensing algorithms then to develop a best performing soft sensors for the data sets. Nonetheless for demonstration purposes, there is, for each experiment, a set of parameters and the predictions of the best performing models included. In the following sections, the notion of optimality is limited to investigated parameter ranges, which does guarantee a globally valid optimality.

### 4.4.1 Applied pre-processing and modelling techniques

At this position, the techniques and their parameter values and ranges for the experiments are described. Parameters stated as ranges, e.g. $[0.1, 0.2 \ldots, 1.0]$, are optimised for each of the experiments.

**Principal Component Analysis (PCA):**

- Origin: Matlab implementation

- Parameters:

  - Number of principal components, expressed as the percentage of variance covered by the principal components: $covVar := [0.85, 0.90, 0.95, 0.99]$

**Robust Principal Component Analysis (robPCA):**

- Origin: robust PCA[2]
- Parameters:
  - Robust scale estimator: $scaleEstim = mad$
  - Number of principal components, expressed as the percentage of variance covered by the principal components: $covVar = [0.85, 0.90, 0.95, 0.99]$

**Multi-Layer Perceptron (MLP):**

- Origin: Netlab toolbox[3]
- Parameters:
  - Number of hidden units: $numHid = [1, 3, 5, 10, 15, 20]$
  - Number of output units: $numOut = 1$
  - Hidden unit(s) activation function : $outActFun = tanh$
  - Output unit activation function : $outActFun = linear$
  - Learning algorithm: The applied learning algorithm is the Scaled Conjugate Gradient (SCG) method (see Section 3.3.2), which is trained over 10000 epochs in order to guarantee its convergence. The rest of the parameters are set to default values (see [131, p. 56] for details).

**Locally Weighted Projection Regression (LWPR):**

- Origin: LWPR[4]
- Parameters: The parameters to be optimised were selected according to the suggestions given in [103]. Parameters not stated in the following listing are set to default values.
  - $initD = [0.1, 1, 10]$
  - $wGen = [0.1, 0.5, 0.75]$
  - $penalty = [10^{-7}, 10^{-6}, 10^{-5}]$
  - $initAlpha = 100$
  - $meta = 1$
  - $diagOnly = 1$
  - $updateD = 0$
  - $kernel =$ 'Gaussian'

[2] http://www.econ.kuleuven.be/public/NDBAE06/programs/#pca
[3] http://www.ncrg.aston.ac.uk/netlab
[4] http://www.ipab.informatics.ed.ac.uk/slmc/software/lwpr

**Local Adaptive Soft Sensing Algorithm (LASSA):**

- Origin: Algorithm proposed in Section 4.3 of this work
- Parameters:
  - Landmarker model type: Linear regression (MLR)
  - Local expert model type: Linear regression (MLR)
  - $n^{init} = [30, 50, 70]$[5] (see Equation 4.1)
  - $\sigma = [10^{-2}, 10^{-3}, 10^{-4}]$ (see Equation 4.11)
  - $\sigma^{adapt} = [10^{-2}, 10^{-3}, 10^{-4}]$ (see Equation 4.19)
- Pre-processing:
  - Robust PCA: $numPrincComp = 0.95$

### 4.4.2 Presentation of the results

In the following experiments the results are presented numerically in form of Mean Squared Errors (MSE) of the predictions and correlation coefficient between the correct target value and the prediction.

The visual presentation of the results is done by using boxplots and sequence plots. The boxplots show the following information:

- Median value: horizontal line inside the boxes
- Upper/Lower quartile: upper/lower edge of the box
- 1.5-times interquartile range: upper/lower end of the whisker
- Outliers: points above/below the whiskers

As such it is a valuable tool for the comparison of the performance achieved by models of varying complexity.

The sequence plots display the correct target variable values next to the predictions of the soft sensors. The x-axis of the sequence plots is labeled as *Time* to indicate that the plots show the temporal sequence of the target variables. Each of the figures is also equipped with a legend showing the MSE and correlation coefficient value of the predictions in the following format: "MSE, corr. coef.".

### 4.4.3 Experiments methodology

The experiments follow the methodology discussed in Section 4.2. Additionally, the following simplifications to the methodology are assumed: (i) the sampling rate of the target variable and the input variable are equal during the on-line prediction phase; (ii) for the adaptive models during the on-line phase, the target variable is delayed by one sample and becomes available after making the predictions for the current sample. This, on one hand, assures that the models are always tested using the out-of-sample principle and, on the other hand, provides the ability to test the adaptation

[5] due to the high dimensionality of the thermal oxidiser data set, the following values had to be used: $[50, 70, 100]$

approaches. This scenario serves well the purpose to analyse the adaptation characteristics of different algorithms.

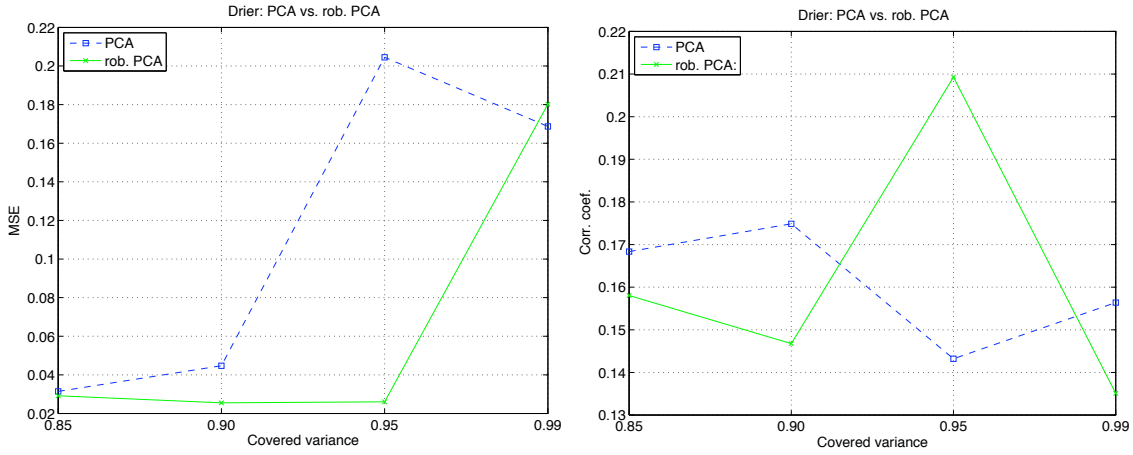
To simulate the historical and on-line data, the available data set is split into two parts. The first 30% of each of the data sets form the historical data, which is used for training the models. The remaining 70% of the data simulate the on-line data and are delivered as a stream of samples. This kind of splitting of the data was chosen in order to be able to assess not only the different soft sensing approaches but more interestingly also the different adaptation mechanisms. Therefore there is more data dedicated to the on-line phase than to the training phase. The only manual data pre-processing for each of the data sets is listed in the corresponding sections of Appendix B.

In the following sections, all of the reported results are based on the evaluations on the independent test data.

### 4.4.4 Industrial drier experiments

#### Non-adaptive soft sensors

**PCA+MLP:** For the non-adaptive PCA+MLP soft sensor the pre-processing methods, PCA and robust PCA, are tested first. The direct comparison of the MSE and correlation coefficient between the PCA and robust PCA can be found in Figure 4.7. The figures present averaged MSE and

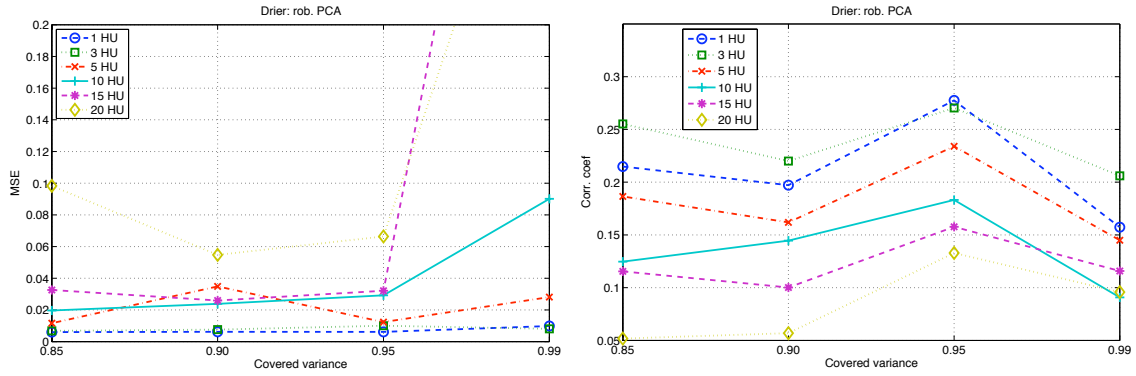



(a) MSE (b) Correlation coefficient

Figure 4.7: Industrial drier: Comparison between the performances achieved with PCA and robust PCA pre-processing for varying settings of variance contained in the principle components, i.e. varying numbers of resulting principle components

correlation coefficients for different values of covered variance, i.e. different numbers of used principal components. Each of the points in the figure is an average value calculated over ten random initialisations of the MLP models and six different MLP topologies, i.e. each of the points is an average over 60 different PCA+MLP models. One can observe the dominance of the robust PCA method, which achieves, for most of the settings, better performance than the traditional PCA algorithm and will therefore be selected for further analysis.

The next step is the evaluation of the influence of the principal components number for the PCA+MLP soft sensor. This is achieved based on the information from Figure 4.8, which shows the MSE and correlation for different parameter settings of the robust PCA+MLP models. The results in Figure 4.8 are again based on the average performance of ten randomly initialised MLPs.




(a) MSE (b) Correlation coefficient

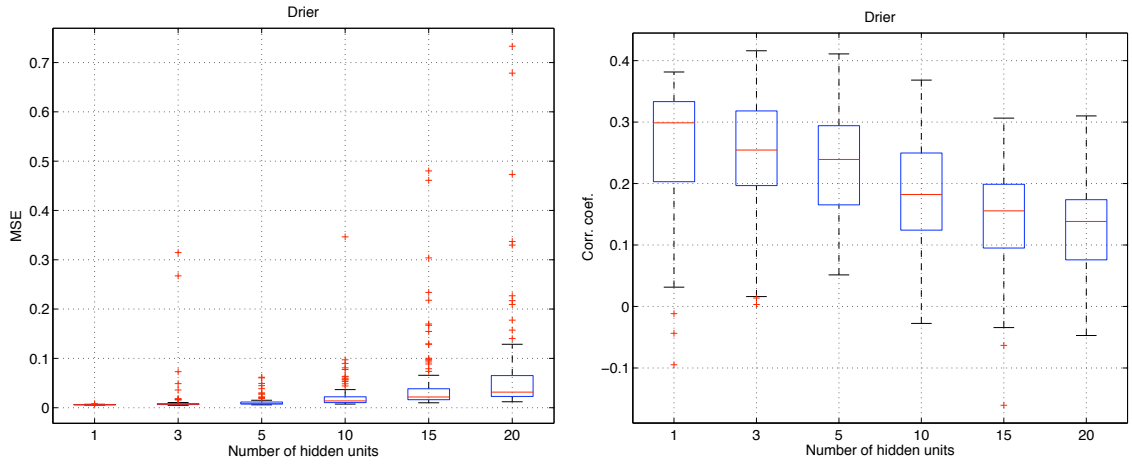
Figure 4.8: Industrial drier: Influence of the variance covered by the robust PCA pre-processing and of the hidden units number on the performance PCA+MLP-based soft sensors

The MSE figure (Figure 4.8(a)) shows more or less stable performance for the first three settings and weaker performance for the coverage of $99\%$ of the variance, which results in high number of principal components. Other than that, it is difficult to make any other decisions for the optimal parameter value. In contrast to this, Figure 4.8(b) provides more interesting information for the three lower settings in terms of the correlation coefficients between the predicted and correct target values. One can observe an increase and a peak of the correlation at $0.95$. Therefore, the chosen optimal pre-processing method for this data set is the robust PCA covering $95\%$ of the variance of the original data set. This corresponds to eight principal components being used for further modelling, which is a significant compression of the data from the original 19 input variables demonstrating the high co-linearity in the raw data set (see Section 2.3.2 for discussion).

In further analysis, the focus is put on analysing the parameter setting of the MLP model. It can already be observed in Figure 4.8 that MLPs with lower number of hidden units achieve better performance. Figure 4.9 can be considered in order to further investigate the performance of the different MLP topologies. The figure shows the statistics of 100 randomly initialised MLP models with a given topology. Considering the boxplots in Figure 4.9, the decision for the optimal complexity is straightforward since the MLP with one hidden unit achieves on average (considering the median performance) the best performance in terms of the MSE and the correlation.

Another fact, which is reflected in the boxplots, is the difficulty of model selection. Even considering the least complex model, which has, in accord with the theory, lowest error variance (see Section 3.2), one can see a large spread of the measured error values (see Figure 4.10 where one can see that the error ranges from ca. $5.5 \cdot 10^{-3}$ to ca. $7.5 \cdot 10^{-3}$). The significance of the difference can be better understood when considering Figure 4.9(b), where it can clearly be seen that the achieved correlation coefficients can be anywhere between $-0.1$ and $0.4$ even for the simplest model. This makes the selection of the best trained model virtually impossible and calls for the application of more advanced approaches. Figure 4.10 also shows a direct effect of the bias/variance trade-off as the variance of the MSE is steadily increasing with the increasing complexity of the model. Due to overfitting, the median MSE is also increasing with the increasing number of hidden units.

Next, the effect of combining the models is going to be analysed. The applied combination method is the average building over ten randomly initialised models. The predictions have the




(a) MSE (b) Correlation coefficient

Figure 4.9: Industrial drier: Performance comparison between PCA+MLP-based soft sensors with varying number of hidden units

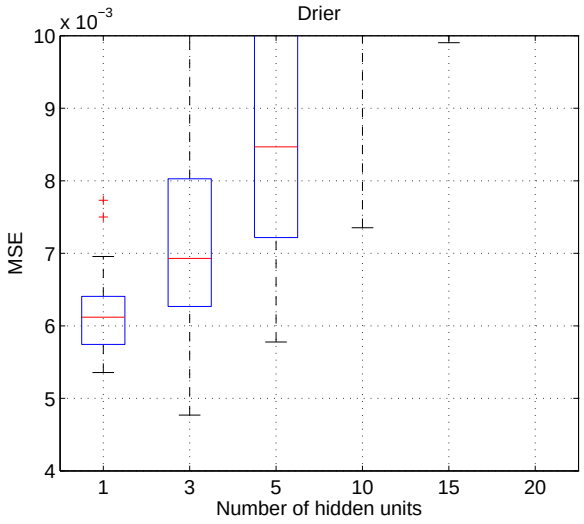
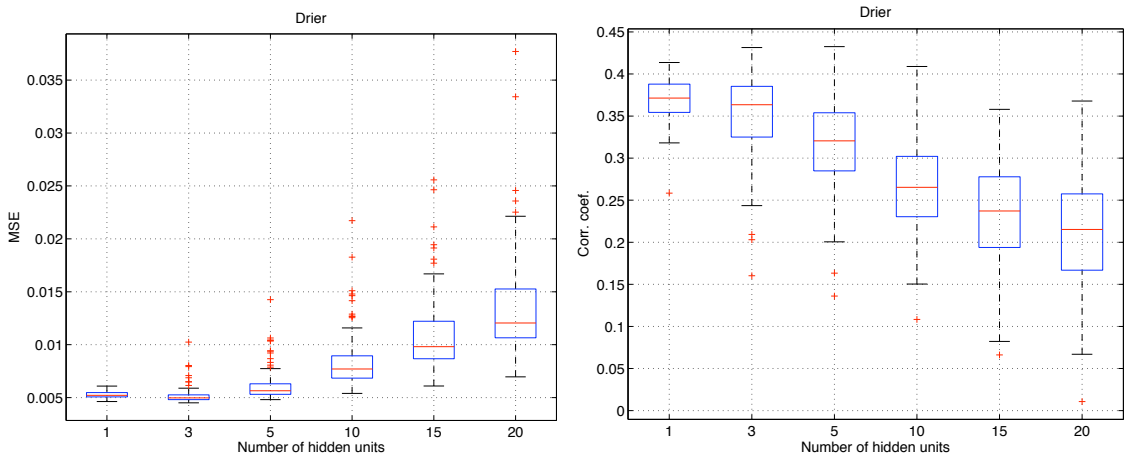



Figure 4.10: Industrial drier: Performance comparison between PCA+MLP-based soft sensors with varying number of hidden units - detailed view

following form:

$$\mathcal{F}(\mathbf{x}) = \frac{1}{10}\sum_{k=1}^{10} f_{(k)}. \tag{4.21}$$

The MSE and correlation coefficient results are presented in Figure 4.11 and Figure 4.12 respectively. At first glance, one can observe a dramatic effect of the combination building on the performance of the models. The average performance of the models improves significantly, which is obvious especially when comparing Figure 4.10 with Figure 4.12. At the same time a reduced variance, reflected in the smaller size of the boxes, is also achieved by applying the simple combination method. Another interesting observation is that the optimal number of hidden units shifts from one hidden unit for the plain models to three hidden units for the combined models. This is




(a) MSE (b) Correlation coefficient

Figure 4.11: Industrial drier: Performance comparison between combined PCA+MLP-based soft sensors with varying number of hidden units

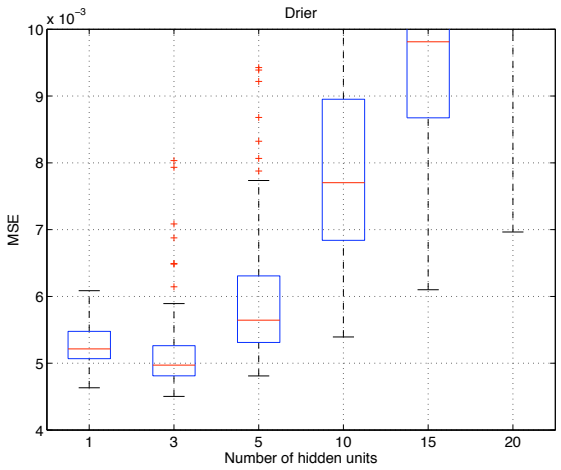



Figure 4.12: Industrial drier: MSE performance comparison between combined PCA+MLP-based soft sensors with varying number of hidden units - detailed view

an effect of the averaging that up to a certain degree, prevents the overfitting of the models to the training data.

All of the above experiments allow one to summarise and select an optimal setting for the PCA+MLP soft sensor:

- Pre-processing technique: robust PCA with $covVar = 0.95$
- Predictive technique: Mean combination of ten MLPs with $numHid = 3$.

The predictions on the test (on-line data) of such a model can be found in Figure 4.13.

**Non-adaptive LWPR:** The next tested soft sensor is based on the LWPR technique. There are three parameters to be analysed in the ranges according to Section 4.4.1. Figure 4.14 shows the boxplot of the parameter sensitivity of the LWPR-based soft sensors. When compared with the

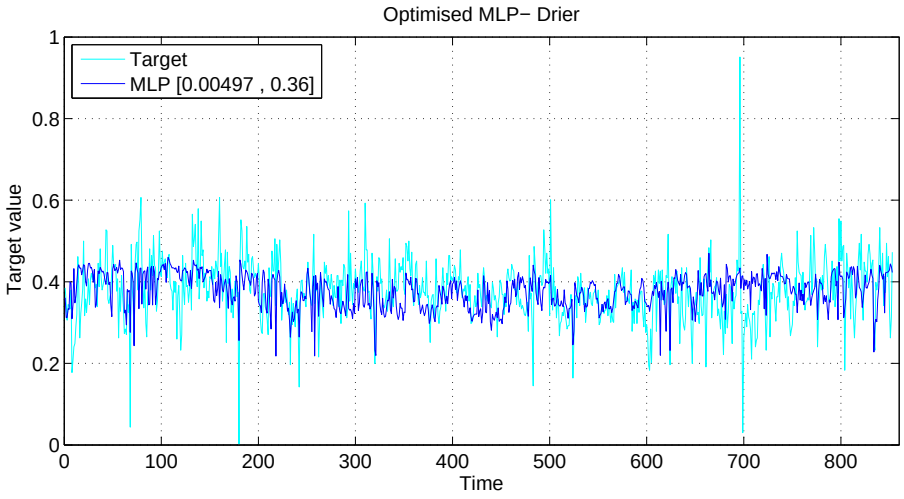



Figure 4.13: Industrial drier: Predictions of the non-adaptive PCA+MLP-based soft sensor

high variance of the MLP-based soft sensors in Figure 4.9(a), the performance variation of the LWPR-based soft sensors appears to be remarkably low. Furthermore, as the LWPR does not suffer from the local minima problem the parameter optimisation is much easier than in the case of the MLP-based soft sensors.

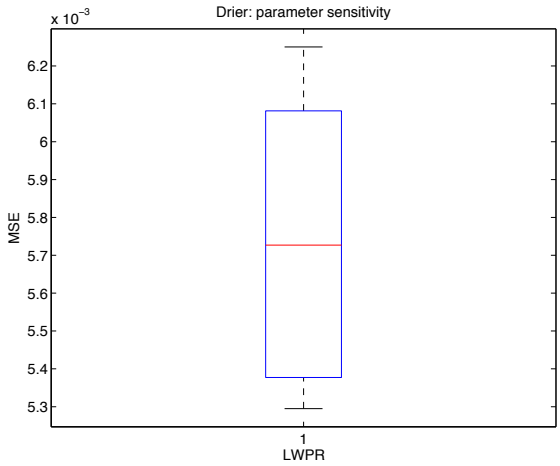



Figure 4.14: Industrial drier: Sensitivity of the LWPR method's performance with respect to its parameter settings

The optimisation process for the non-adaptive LWPR-based soft sensor resulted in the following optimal settings:

- $initD = 1$
- $wGen = 0.1$
- $penalty = 10^{-7}$

The model resulting from the provided training data consists of a single receptive field. The predictions of the model are shown in Figure 4.15.

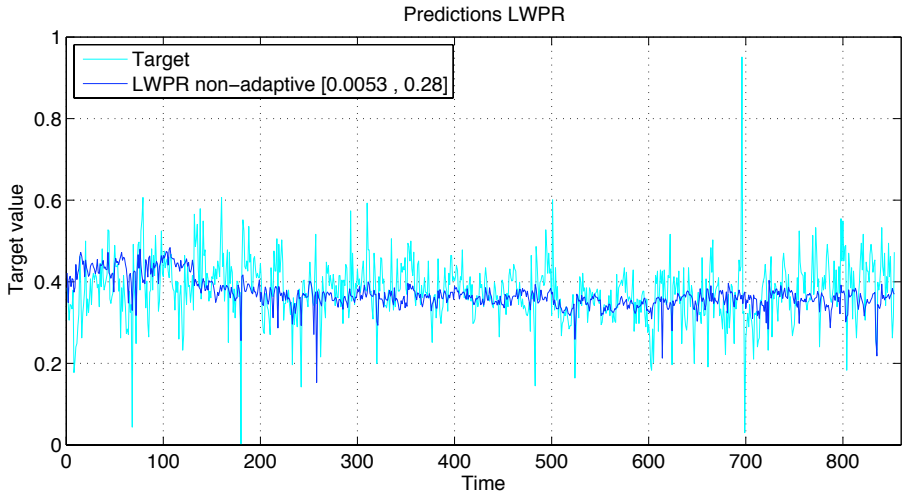
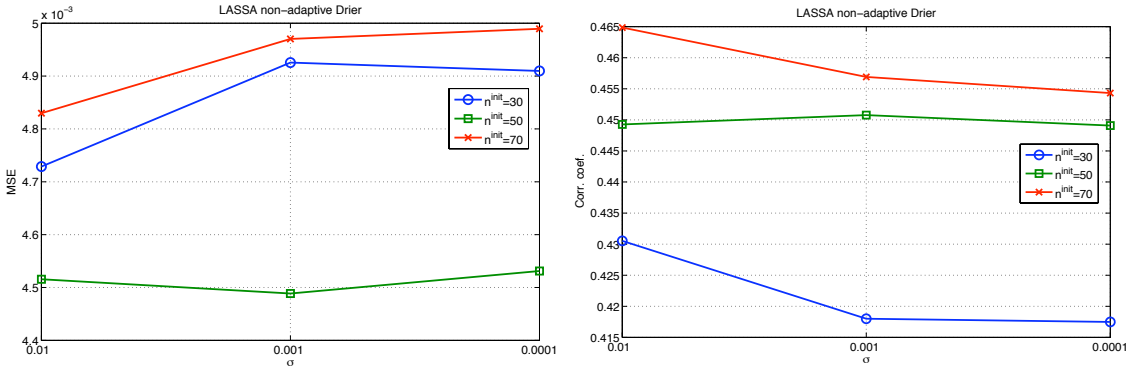



Figure 4.15: Industrial drier: Predictions of the non-adaptive LWPR-based soft sensor

**Non-adaptive LASSA:** There are two parameters of the method that need to be looked at, these are (i) $\sigma$: the variance of the Gaussian functions for the building of the local expert descriptors $\mathcal{L}$; and (ii) $n^{init}$: the size of the initial window for the receptive field estimation. The MSE and correlation coefficients for different settings of the parameters are shown in Figure 4.16. The




(a) MSE (b) Correlation coefficient

Figure 4.16: Industrial drier: Influence of $n^{init}$ and $\sigma$ on the performance of the non-adaptive LASSA-based soft sensor

figures show that the sensitivity of the performance due to the values of the input parameters is quite low and the soft sensor performs well for all combinations of the parameter values. This fact is also reflected in Figure 4.17, which summarises the MSE performances of all of the tested parameter settings. The figure shows that, compared to the LWPR-based (see Figure 4.14) and PCA+MLP-based (see Figure 4.11) soft sensors, the MSE performance of the LASSA-based soft sensors is higher and shows lower fluctuation at the same time.

The following values have been selected as optimal:

- $\sigma = 10^{-3}$
- $n^{init} = 50$.

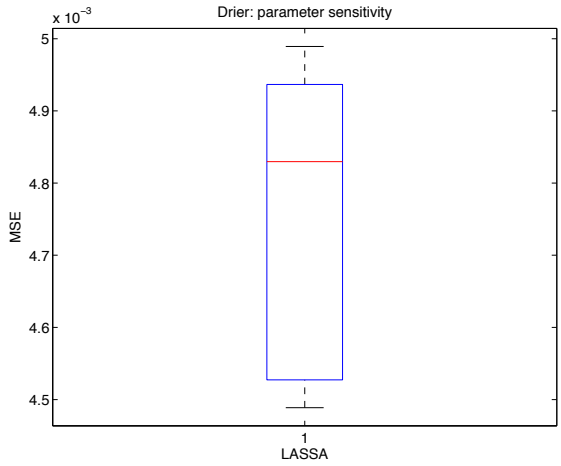



Figure 4.17: Industrial drier: Sensitivity of the LASSA method's performance with respect to its parameter settings

These parameter values resulted in building five receptive fields for the training data. For each of the receptive fields a local PCA pre-processing is applied, for which the number of principal components varies between six and eight. Figure 4.18 shows the predictions and combination weights of the local experts. One can observe the diversity of the predictions and that none of the local experts is able to deliver accurate predictions for the whole test data set on its own. The combined prediction can be found in Figure 4.19.

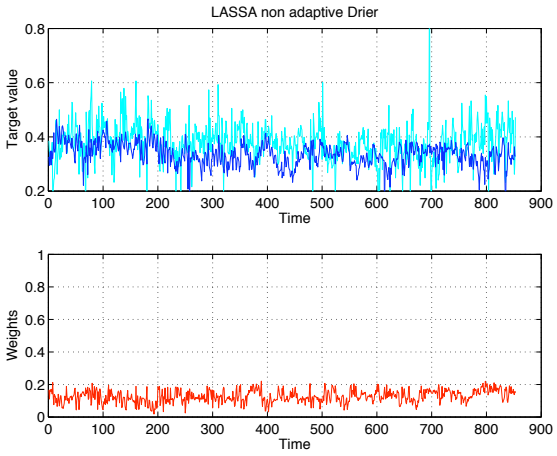
Comparing the predictions of the soft sensor shown in Figure 4.19 to the previous two non-adaptive soft sensors, i.e. LWPR- and PCA+MLP-based, it is obvious that it outperforms the non-adaptive LWPR-based soft sensor shown in Figure 4.15 as well as the median performance of the PCA+MLP-based models (see Figure 4.12). In fact, the achieved performance is even outside of the boxes of PCA+MLP-based soft sensors, as can be seen in Figures 4.11(a) and 4.11(b) for the MSE and correlation coefficient respectively.

### Adaptive soft sensors

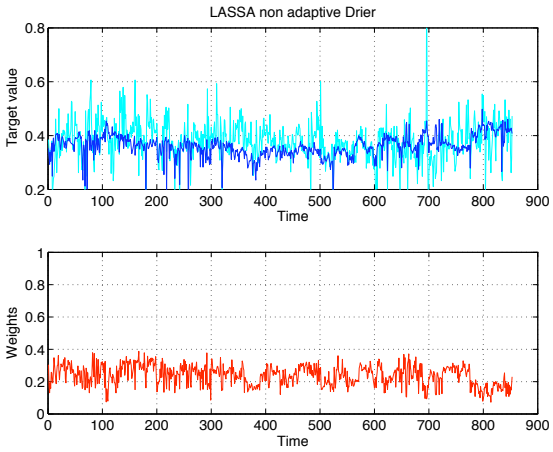
**PCA+MLP with moving window adaptation:** In this section the first experiments with adaptive soft sensors are carried out. The first experiment applies the Moving Window (MW) technique combined with PCA+MLP with the optimal parameter setting. In general the MW technique has two parameters: (i) the size of the window; and (ii) the step size, i.e. the intervals in which the model is retrained. For the presented experiments, the window size is kept constant (equal to the size of the training data) and only the influence of the step size on the performance is studied.

The influence of retraining the model using the moving window technique can be observed in Figure 4.20. Contrary to the expected performance improvement, the performance drops for all step sizes. This indicates that model adaptation does not necessarily lead to better performance and has to be applied carefully only in situations where it is actually required.

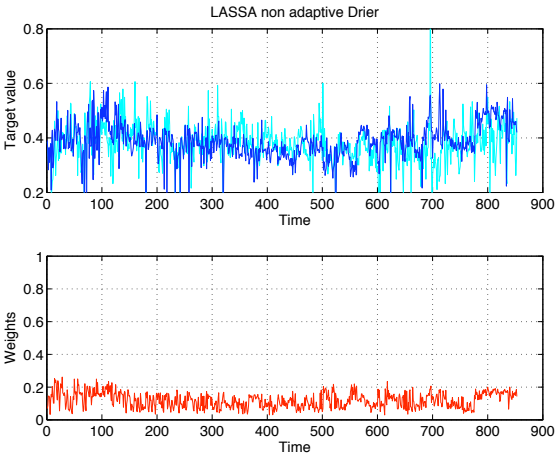
**Adaptive LWPR:** Compared to the non-adaptive case (see Figure 4.14), the parameter sensitivity of the adaptive LWPR technique shown in Figure 4.21 is lower and the performance level improves at the same time. This is an evidence of the effectiveness of the adaptation mechanisms.



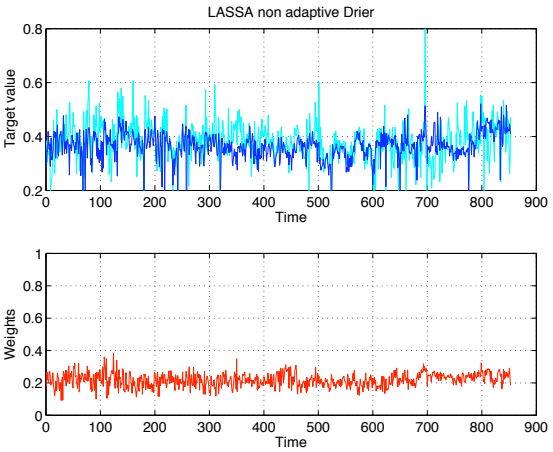

(a) 1. local expert



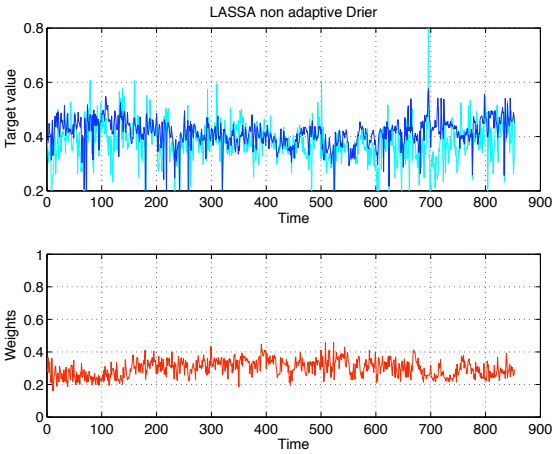

(b) 2. local expert




(c) 3. local expert




(d) 4. local expert




(e) 5. local expert

Figure 4.18: Industrial drier: Predictions and combination weights $v_k$ of five local experts

The optimal parameters for the adaptive version of the LWPR soft sensor are slightly different

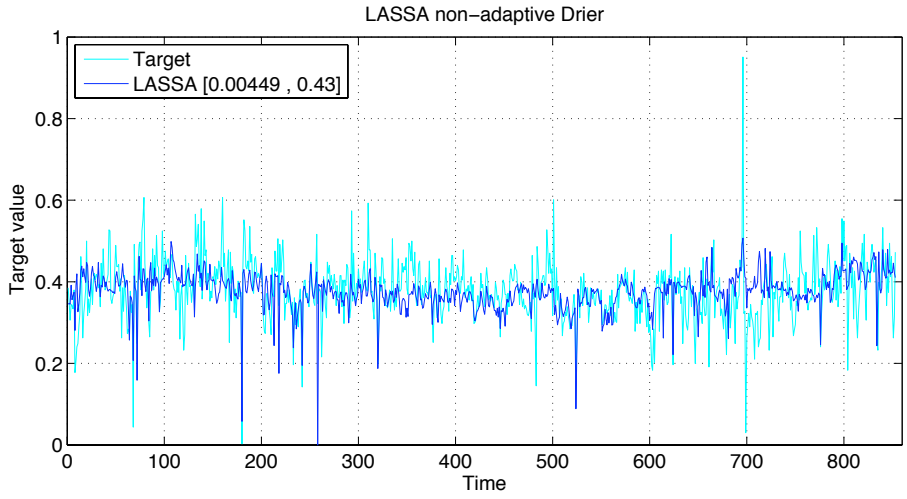
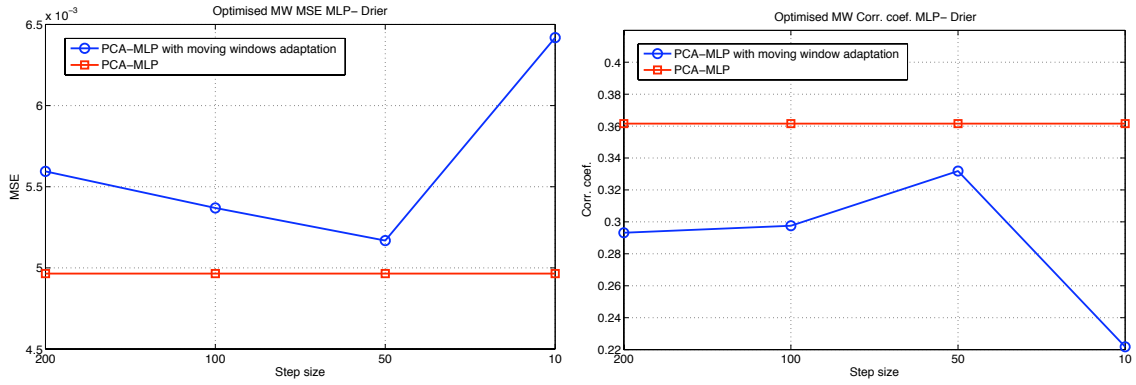



Figure 4.19: Industrial drier: Predictions of the non-adaptive LASSA-based soft sensor




(a) MSE (b) Correlation coefficient

Figure 4.20: Industrial drier: Effect of the moving window step size on the performance of the adaptive PCA+MLP-based soft sensor

to the non-adaptive version:

- $initD = 1$
- $wGen = 0.75$
- $penalty = 10^{-7}$.

As the training continues incrementally on the on-line data, new receptive fields are continuously deployed. The predictions of this model shown in Figure 4.22 show an improvement compared to its non-adaptive version, demonstrating the effectiveness of this incremental algorithm.

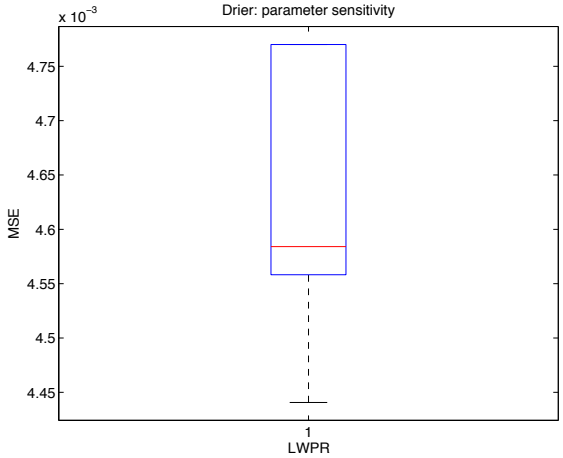



Figure 4.21: Industrial drier: Sensitivity of the LWPR method's performance with respect to its parameter settings

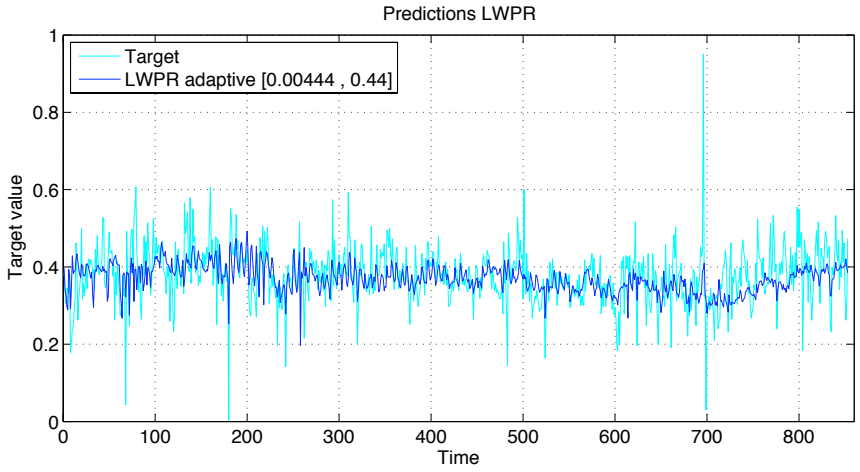


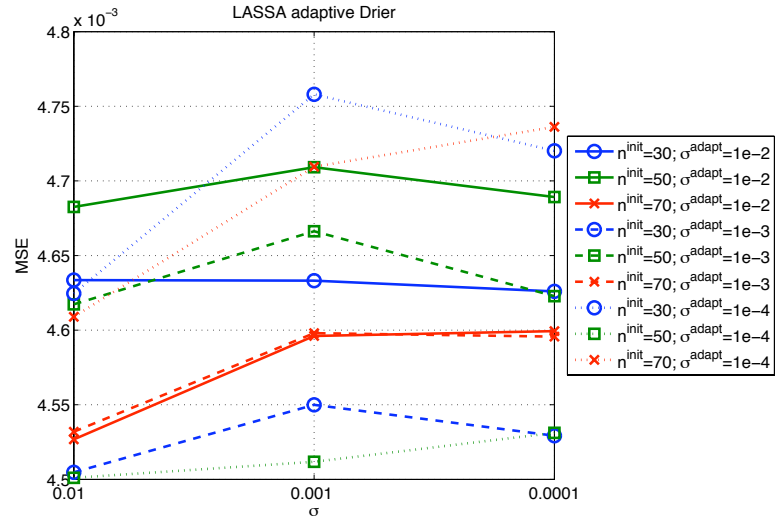

Figure 4.22: Industrial drier: Predictions of the adaptive LWPR-based soft sensor

**Adaptive LASSA:** Next, the performance of the adaptive LASSA-based soft sensor is assessed. Figure 4.23 allows one to analyse different values of the three input parameters and the performance achieved by the soft sensors with corresponding parameter values. There is an contradiction between the MSE and correlation coefficient plots. For the correlation, the highest values, i.e. best performance, is achieved by the three models with large initial windows ($n^{init} = 70$), but in terms of the MSE only two of these models perform well.

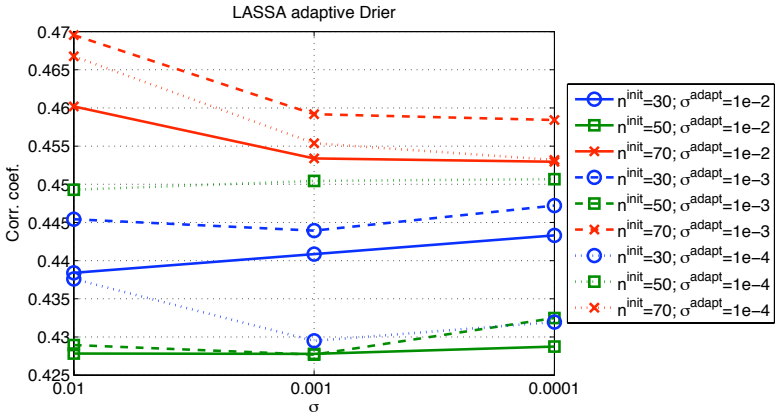
Taking the figure into account, it is possible to conclude that the performance variation between the different settings is rather low, which shows a certain robustness of the algorithm towards the parameter settings. This is also confirmed in Figure 4.24. Although the range of the MSE does not change a lot, a shift of the median performance can be observed between the non-adaptive and adaptive case (compare Figure 4.17 with Figure 4.24).

Based on the previous evidence, the following combination can be perceived as optimal:

- $\sigma = 10^{-2}$




(a) MSE




(b) Correlation coefficient

Figure 4.23: Industrial drier: Influence of $n^{init}$, $\sigma$ and $\sigma^{adapt}$ on the performance of the adaptive LASSA-based soft sensor

- $\sigma^{adapt} = 10^{-3}$
- $n^{init} = 50$.

The soft sensor with these settings can be further analysed by considering Figure 4.25, which shows the predictions of the particular local experts and their associated combination weights. The above settings lead to the deployment of five receptive fields. Considering the weights in Figure 4.25, one can see that they correlate well with the validity of the local experts, e.g. the first local expert demonstrates this fact nicely. Another effect that can be observed is the influence of the adaptation mechanisms. In the non-adaptive case, shown in Figure 4.18, one can see that the weights were *flat* with a low fluctuation, which is not the case when the adaptation is applied.

Finally, Figure 4.26 shows the predictions of the soft sensor based on the adaptive LASSA approach.

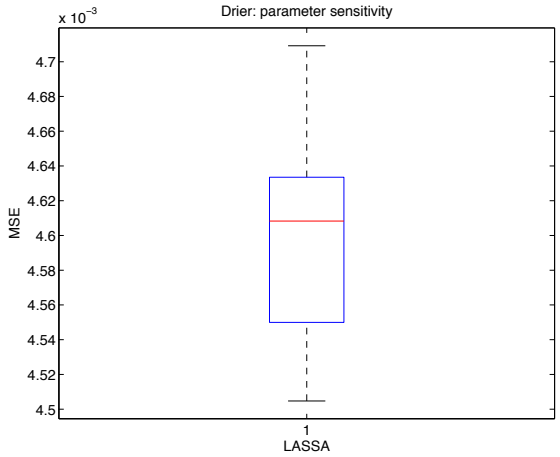



Figure 4.24: Industrial drier: Sensitivity of the LASSA method's performance with respect to its parameter settings

| | Evonik's soft sensor | LASSA soft sensor |
|---|---|---|
| MSE | $2.88 \cdot 10^{-2}$ | $4.39 \cdot 10^{-3}$ |
| Corr. Coeff. | 0.32 | 0.43 |

Table 4.1: Industrial drier: Comparison of the MSE and correlation coefficient performances between the non-adaptive soft sensor implemented at Evonik and the non-adaptive LASSA-based soft sensor

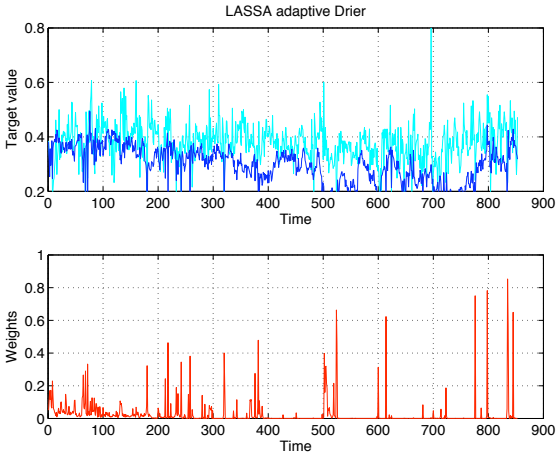
### Evonik's soft sensors

In this section, both LASSA soft sensors (non-adaptive and adaptive) are compared to the soft sensor developed, implemented and used at Evonik Degussa GmbH. Although the same data set was used for the development of Evonik's soft sensors, the data were split in a different way (ca. $68\%$ of the data used for training and $32\%$ for test data) at Evonik. In order to be able to compare the performance of the soft sensors objectively, the predictions were compared over the same test data range.

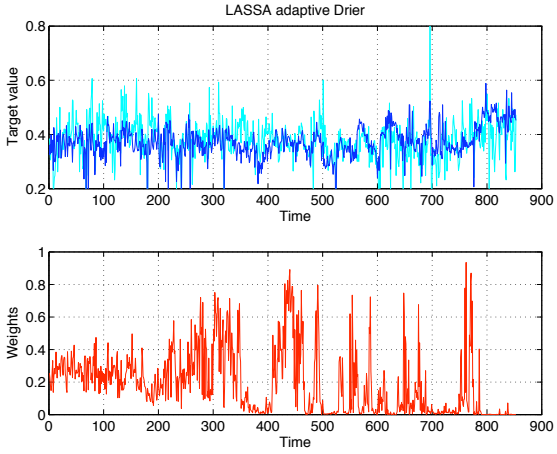
There were two different soft sensors developed at Evonik, the first one is a conventional PLS-based non-adaptive soft sensor (further details of the soft sensor cannot be presented for confidentiality reason). The second one is an adaptive moving window PLS-based soft sensor developed and presented in [168].

**Non-adaptive version:** The operation of the non-adaptive versions of Evonik's and LASSA soft sensors over the period of almost four months is shown in Figure 4.27. The figure shows that both soft sensors work well over the first half of the data. For the second half the Evonik soft sensor develops an offset and its predictions are no longer accurate. As for the LASSA-based soft sensor, it is able to provide accurate predictions over the whole range presented in Figure 4.27.

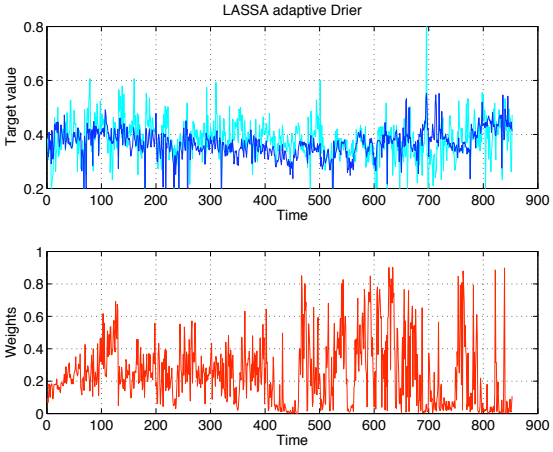
The issue of the Evonik's soft sensor is also projected into Table 4.1, which compares the performances of the non-adaptive models. The table shows that the LASSA soft sensor outperforms the Evonik's soft sensor.



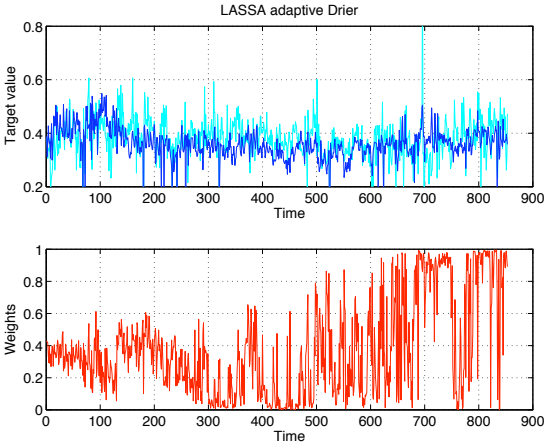

(a) 1. local expert



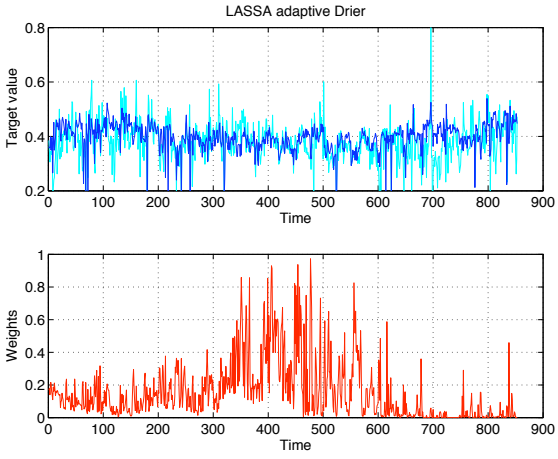

(b) 2. local expert




(c) 3. local expert




(d) 4. local expert




(e) 5. local expert

Figure 4.25: Industrial drier: Predictions and combination weights $v_k$ of five local experts

**Adaptive version:** As mentioned above, for the adaptive soft sensor implementation the moving window technique was applied. The optimal window size for the sensor is ca. 200 samples long

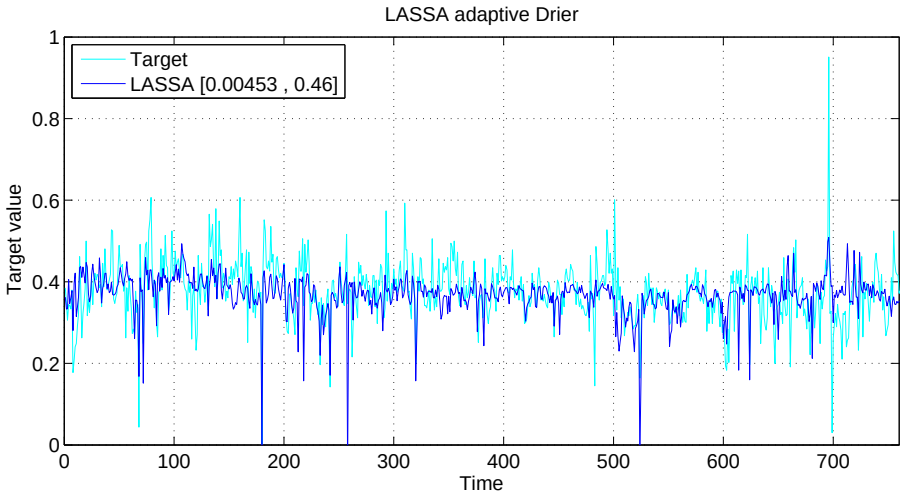



Figure 4.26: Industrial drier: Predictions of the adaptive LASSA-based soft sensor

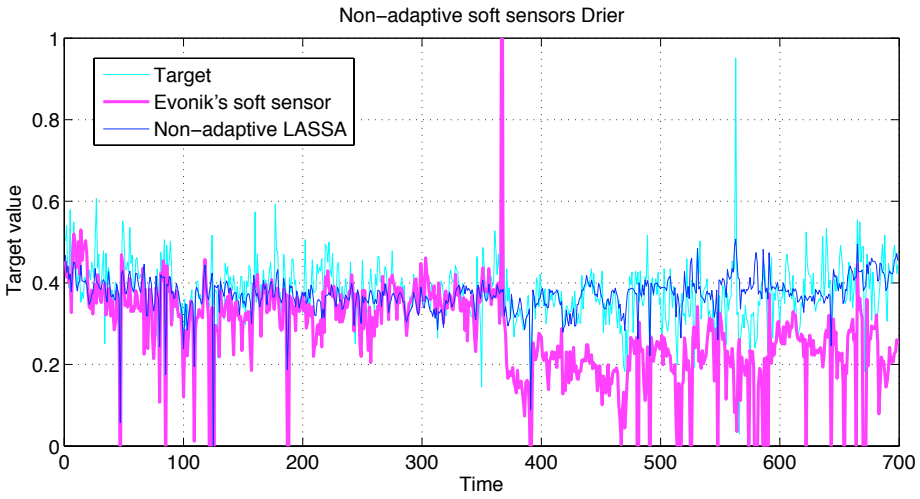



Figure 4.27: Industrial drier: Comparison between a PLS-based soft sensor implemented at Evonik and the non-adaptive version of LASSA-based soft sensor

and the model is used to predict the next ca. 80 samples. After this prediction, the 80 test samples are included into the model and the next 80 samples are used as test data. The prediction of this soft sensor, together with the adaptive version of the LASSA-based soft sensor, are presented in Table 4.2. One can observe that the adaptive Evonik's soft sensor can incorporate the changes of the process and provides useful predictions for the entire target variable.

Table 4.2 shows that the performances of both adaptive soft sensors are similar. Since the adaptive Evonik soft sensor was found to be useful by Evonik's soft sensor developers, this table also validates the LASSA (and LWPR) based soft sensors as these achieve comparable or better performance.

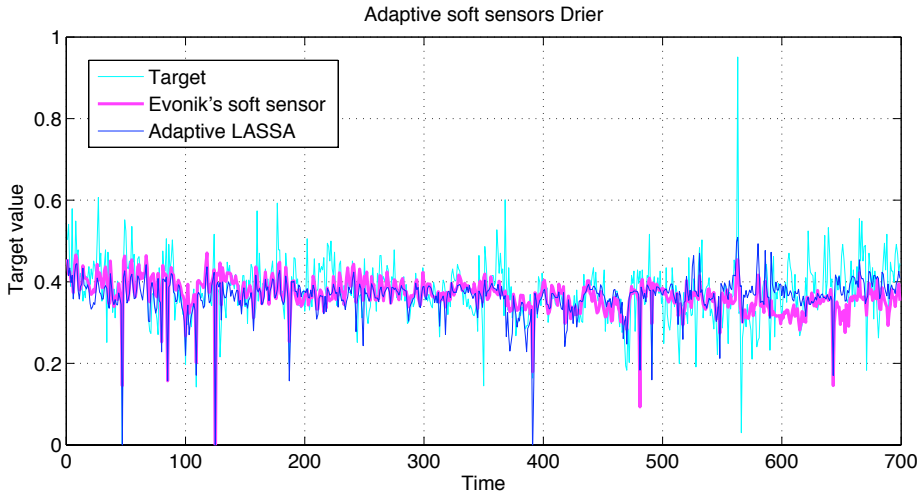



Figure 4.28: Industrial drier: Comparison between the adaptive PLS-based soft sensor implemented at Evonik and the adaptive LASSA-based soft sensor.

| | Evonik's soft sensor | LASSA soft sensor |
|---|---|---|
| MSE | $4.56 \cdot 10^{-3}$ | $4.50 \cdot 10^{-3}$ |
| Corr. Coeff. | 0.42 | 0.45 |

Table 4.2: Industrial drier: Comparison of the MSE and correlation coefficient performances between the adaptive soft sensor implemented at Evonik and the adaptive version of LASSA

### Industrial drier experiments summary

The analysis of this data set has revealed some interesting facts. The experiments with MLP-type soft sensors have shown the benefit of applying a robust version of the PCA algorithm. Applying a simple ensemble method, namely the average over ten models, has proven to be beneficial for two reasons. On one hand the error variance of the plain MLP models has been strongly reduced and on the other hand the accuracy of the combined MLP soft sensors is better. Another interesting effect that was observed is that the ensemble method prevents overfitting and thus allows one to apply more complex models achieving higher performance.

It was also demonstrated that the performance of the non-adaptive as well as the adaptive version of the LWPR algorithm was much more stable in respect of its parameter settings. The performance values achieved using the different settings were also better than those of the PCA+MLP-based soft sensors, which makes this method interesting for soft sensor development.

As for the LASSA-based soft sensor proposed in this work, the performance variation due to the different parameter settings was also very low and the achieved performance levels were the best for the non-adaptive models and comparable to those of the LWPR-based soft sensors in the adaptive case.

The LASSA-based soft sensors were also compared to soft sensors developed and applied at Evonik Degussa GmbH. The comparison has shown that the non-adaptive version of LASSA clearly outperforms the PLS-based soft sensor implemented at Evonik. In the case of the adaptive versions, the LASSA and Evonik's moving window-based soft sensor deliver similar performances.

Another conclusion that can be drawn is that, for this data set, there is no particular need for

| | non-adaptive | | | adaptive | | |
|---|---|---|---|---|---|---|
| | PCA+MLP | LWPR | LASSA | PCA+MLP+MW | LWPR | LASSA |
| MSE | $4.97 \cdot 10^{-3}$ | $5.30 \cdot 10^{-3}$ | $\mathbf{4.49 \cdot 10^{-3}}$ | $5.19 \cdot 10^{-3}$ | $\mathbf{4.33 \cdot 10^{-3}}$ | $4.53 \cdot 10^{-3}$ |
| Corr. Coeff. | 0.36 | 0.28 | **0.43** | 0.33 | 0.45 | **0.46** |

Table 4.3: Industrial drier: Comparison of the MSE and correlation coefficient performances between the different soft sensing approaches

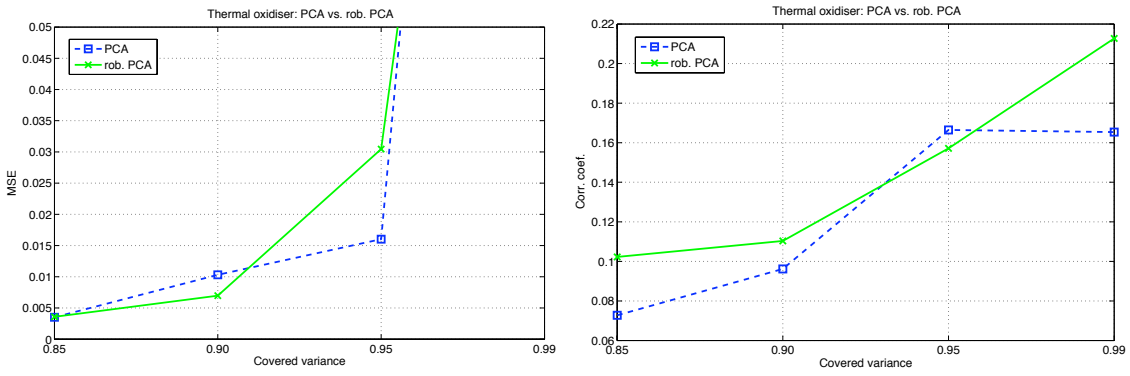
adaptive mechanisms since there is no large gap between the best non-adaptive and adaptive soft sensors. In fact, applying an adaptation method can even prove harmful for the models as shown in the case of the moving window-based adaptation.

The conclusions stated here are also reflected in Table 4.3, which summarises the performances of the tested models.

### 4.4.5 Thermal oxidiser experiments

#### Non-adaptive soft sensors

**PCA+MLP:** We again begin the analysis with the comparison between the conventional and the robust PCA pre-processing algorithms. In the case of this data set, the difference between the two PCA versions is not as obvious as for the industrial drier data. Nevertheless, a small benefit of the robust PCA can still be observed in Figure 4.29 and therefore for further analysis this pre-processing method is going to be applied.

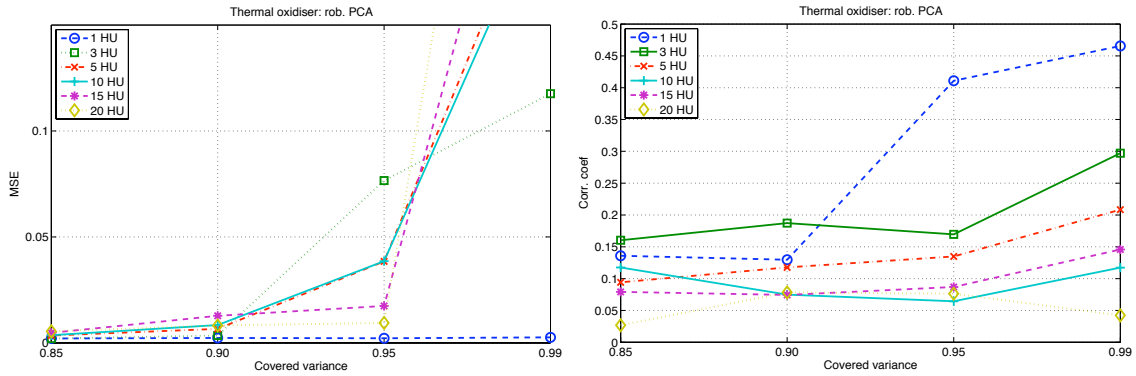



(a) MSE (b) Correlation coefficient

Figure 4.29: Thermal oxidiser: Comparison between the performances achieved with PCA and robust PCA pre-processing

In the next step, the role of the amount of covered variance for the robust PCA is evaluated. Considering Figure 4.30, one can see that the squared error starts to increase for parameter values larger than $0.90$. The correlation coefficient plot shows that the level of correlation remains stable for the first three parameter settings and slightly increases for the largest amount of covered variance (with the exception of one hidden unit MLPs). Considering the plots, it can be concluded that the optimal parameter value for the robust PCA is $90\%$ of covered variance.

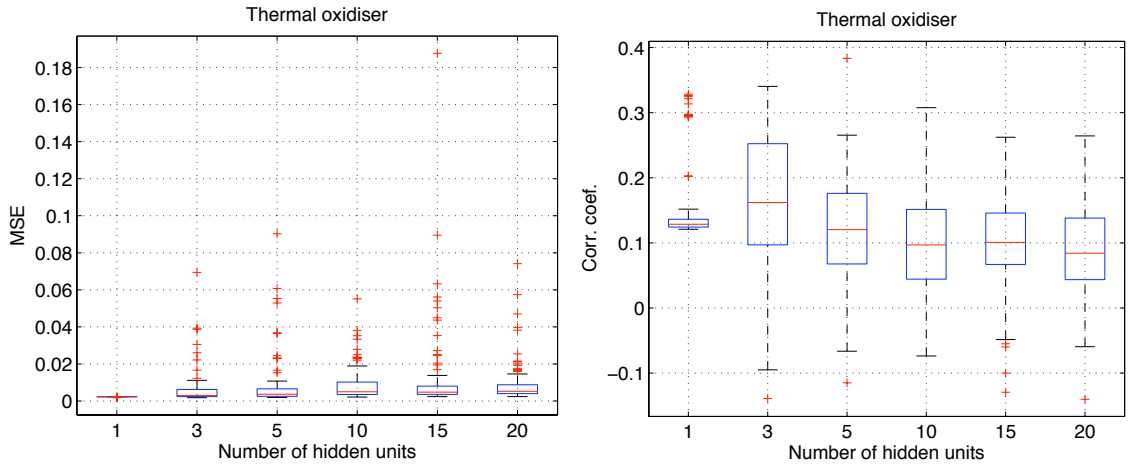
After optimising the pre-processing, the next step is analysing the influence of different settings of the predictive technique. This goal can be achieved by focusing on Figures 4.31 and 4.33 which shows statistics of the MSE/correlation coeffiecient values of the plain MLPs and the combined




(a) MSE (b) Correlation coefficient

Figure 4.30: Thermal oxidiser: Influence of the variance covered by the robust PCA pre-processing and of the hidden units number on the performance PCA+MLP-based soft sensors

models respectively.




(a) MSE (b) Correlation coefficient

Figure 4.31: Thermal oxidiser: Performance comparison between PCA+MLP-based soft sensors with varying number of hidden units

The most obvious observation is the dramatic decrease in the variance of the MSE, while in Figure 4.31 the most extreme outliers of the MSE performance are around $1.00 \cdot 10^{-1}$, for the combined MLPs the most extreme performance outliers are bellow $6.5 \cdot 10^{-3}$. Comparing Figure 4.32 and 4.33(a), which show the MSE boxplots in the same ranges, one can also see the strong improvement in the median MSE values achieved by combining the predictions. Considering both parts of Figure 4.33, the optimal number of hidden units is three.

Summarising the above observations, the optimal PCA+MLP model for this data set has the following parameters:

- Pre-processing: robust PCA with $covVar = 0.90$
- Predictive technique: mean combination of ten MLPs with $numHid = 3$.

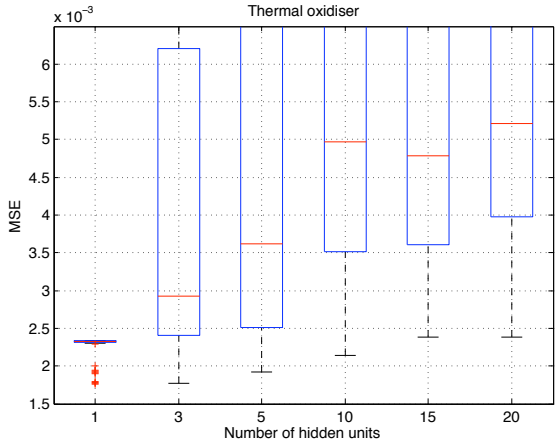
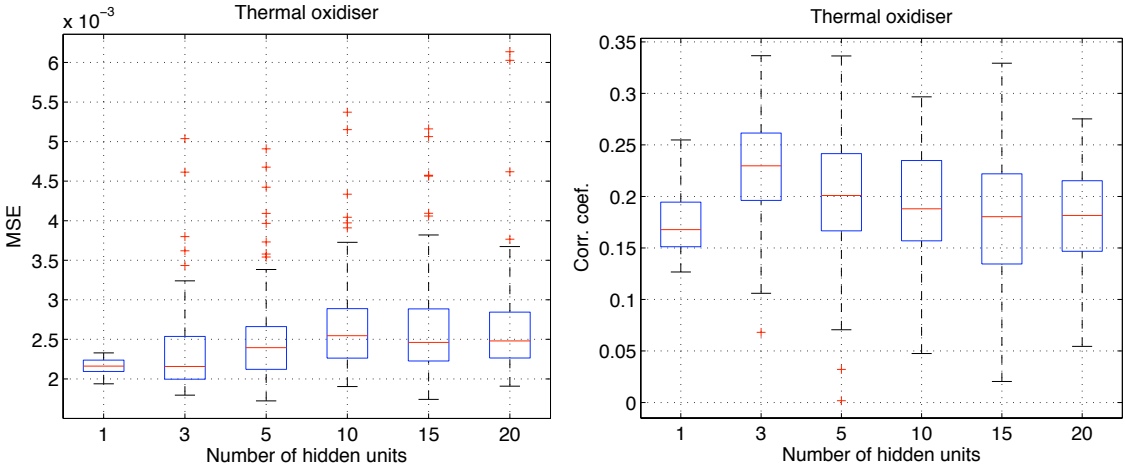



Figure 4.32: Thermal oxidiser: MSE performance comparison between PCA+MLP-based soft sensors with varying number of hidden units - detailed view




(a) MSE (b) Correlation coefficient

Figure 4.33: Thermal oxidiser: Performance comparison between combined PCA+MLP-based soft sensors with varying number of hidden units

The predictions of such a soft sensor can be found in Figure 4.34. The performance is not particularly strong which hints at the need for adaptation.

**Non-adaptive LWPR:** Figure 4.35 shows the performance variation caused by the different parameter settings of the LWPR algorithm. It can be observed that there is a very high variance of the performance in the case of this data. Nevertheless, the median value is similar to the median values of the combined PCA+MLP models shown in Figure 4.33.

The optimisation on this data set led to the following parameters:

- $initD = 0.1$
- $wGen = 0.1$
- $penalty = 10^{-7}$

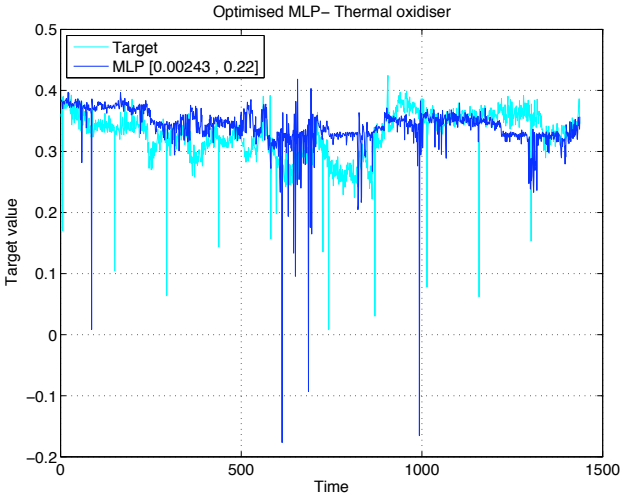



Figure 4.34: Thermal oxidiser: Predictions of the non-adaptive PCA+MLP-based soft sensor

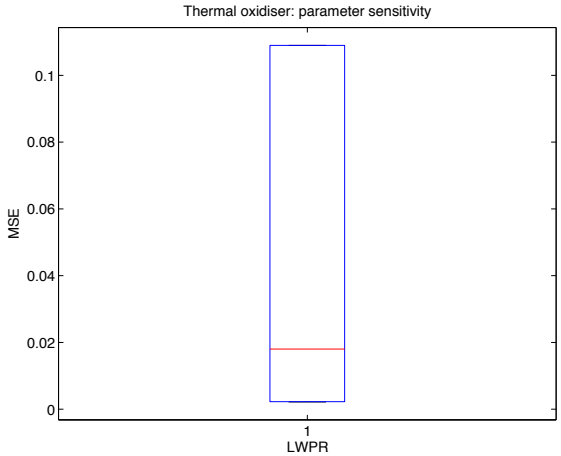



Figure 4.35: Thermal Oxidiser: Sensitivity of the LWPR method's performance with respect to its parameter settings

The predictions of the optimally parametrised model are shown in Figure 4.36. The figure shows that the soft sensor has difficulties following the trend of the data, and there is also a need for model adaptation.

**Non-adaptive LASSA:** The first step of the LASSA-based soft sensor development is again the the analysis of the influence of $\sigma$ and $n^{init}$, which is presented in Figure 4.37. In this case, the selection of the optimal values is relatively simple because the parameter combination $\sigma = 10^{-2}$ and $n^{init} = 50$ leads to the best performance in terms of MSE as well as correlation coefficient.

The sensitivity of the model performance with respect to the parameter settings, summarised in Figure 4.38, is very low (compare e.g. to the one of the LWPR method in Figure 4.35). At the same time, the achieved performance values are the best of all tested non-adaptive models.

Using the optimal settings results in the construction of 15 receptive fields for the training data. The predictions of the non-adaptive LASSA-based soft sensor are shown in Figure 4.39.

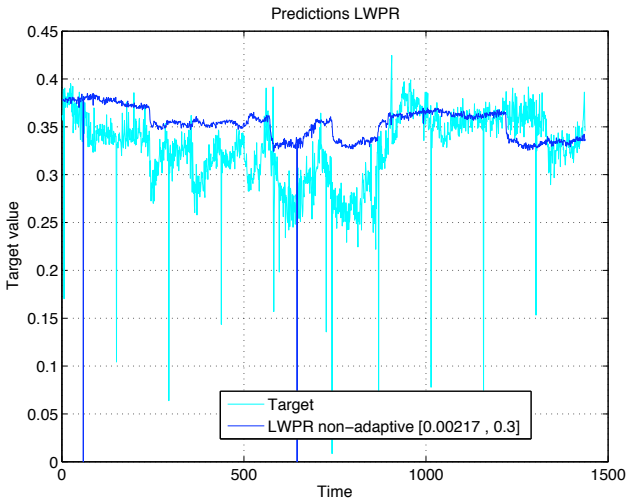
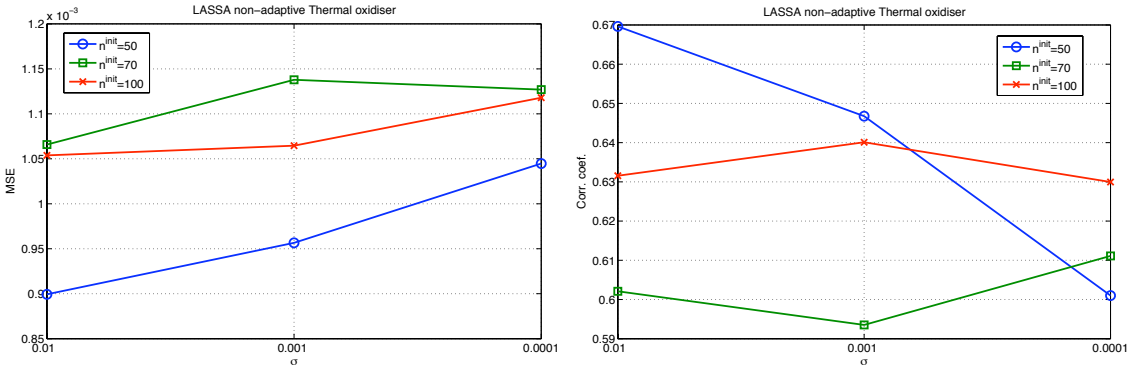



Figure 4.36: Thermal oxidiser: Predictions of the non-adaptive LWPR-based soft sensor




(a) MSE (b) Correlation coefficient

Figure 4.37: Thermal oxidiser: Influence of $n^{init}$ and $\sigma$ on the performance of the non-adaptive LASSA-based soft sensor

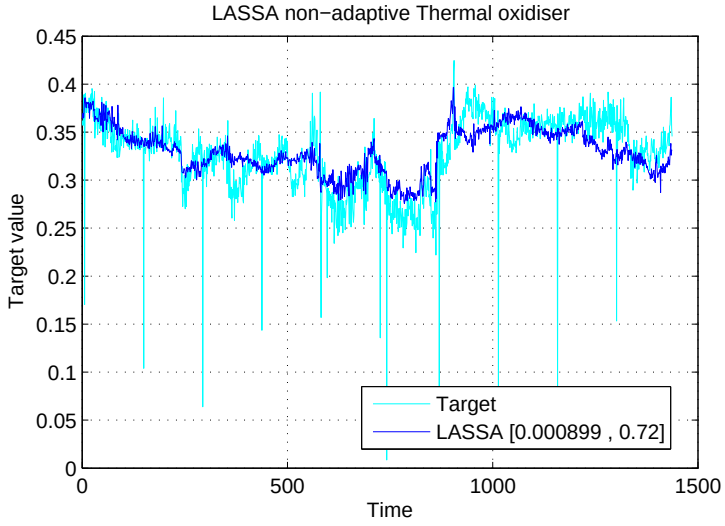



Figure 4.39: Thermal oxidiser: Predictions of the non-adaptive LASSA-based soft sensor

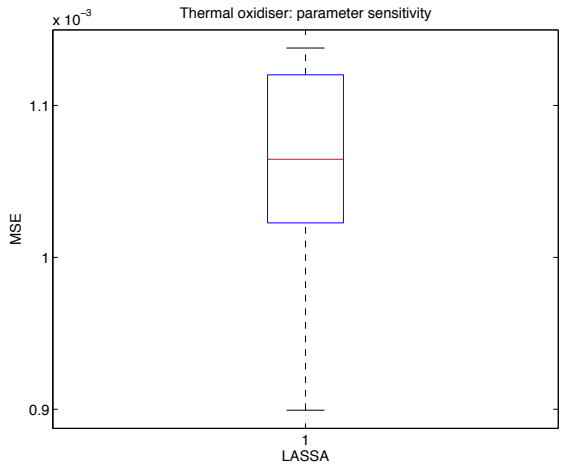
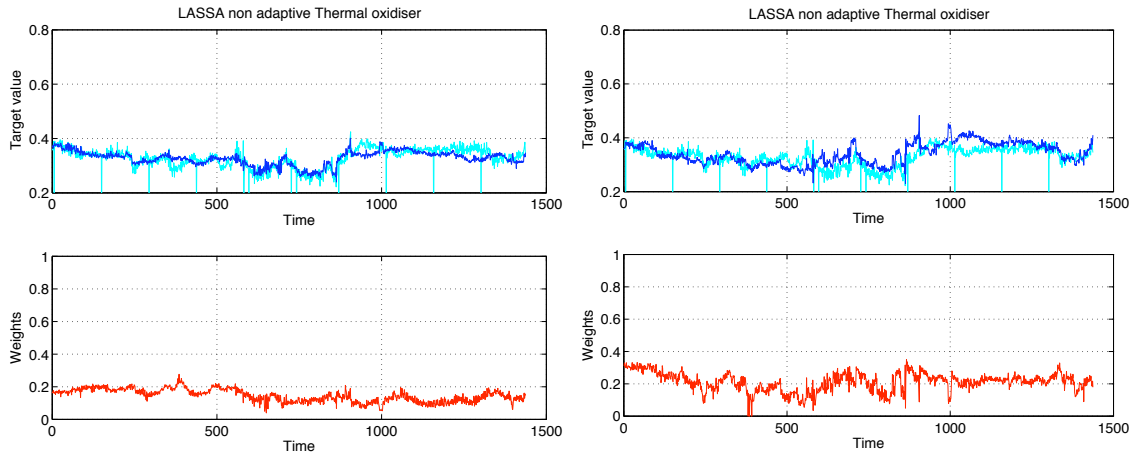



Figure 4.38: Thermal oxidiser: Sensitivity of the LASSA method's performance with respect to its parameter settings

One can see that this soft sensor is performing best from the considered soft sensor types. This is due to the diversity of the local experts and their ability to cover different parts of the on-line data as demonstrated in Figure 4.40. This figure shows two examples of local experts, one of them covering approximately the first 800 samples of the data and the other one performing well for the last 200 data samples.

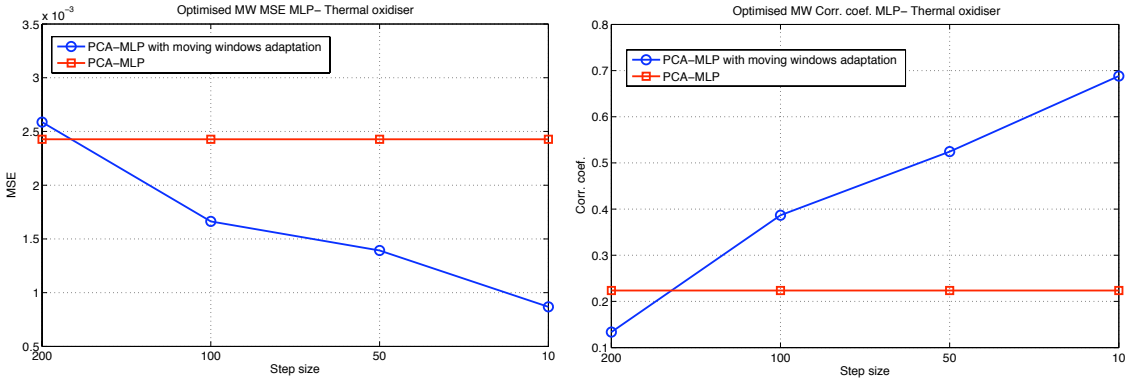



(a) 6. local expert (b) 2. local expert

Figure 4.40: Thermal oxidiser: Two local experts with different areas of expertise

### Adaptive soft sensors

**PCA+MLP with moving window adaptation:** The influence of the retraining of the PCA+MLP soft sensor using the moving window technique can be observed in Figure 4.41. For this data set we can observe a steady improvement in the performance using this simple adaptation technique (with the exception of the largest step size). For the smaller step sizes especially, the increase in performance is quite high for both MSE as well as the correlation. This fact indicates




(a) MSE (b) Correlation coefficient

Figure 4.41: Thermal oxidiser: Effect of the moving window step size on the performance of the adaptive PCA+MLP-based soft sensor

that, contrary to the previous data set, in this case there is a benefit from the adaptation of the models.

To demonstrate this fact, Figure 4.42 shows the prediction of the soft sensor adapted with the four different step sizes.

**Adaptive LWPR:** Applying the adaptation technique of the LWPR algorithm did not have any significant effect on the performance sensitivity of the resulting models and range of the achieved MSE values remained more or less similar to the non-adaptive case (compare Figure 4.35 with Figure 4.43). However, there was an improvement in the median performance which decreased from $1.80 \cdot 10^{-2}$ in the non-adaptive case to $0.46 \cdot 10^{-2}$.

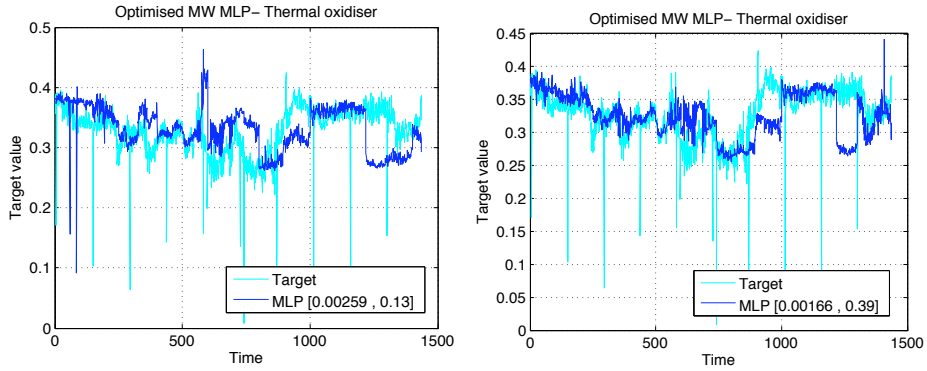
The optimal parameters for the adaptive version of the LWPR soft sensor for this data set are:

- $initD = 0.1$
- $wGen = 0.5$
- $penalty = 10^{-7}$

The results achieved using this adaptive model are reported in Figure 4.44. Similar to the previous data set, a performance increase of the adaptive model can be observed. This soft sensor has no problem with following the trends of the data, although the switching between the two different states of the data seems to be slightly delayed, which is particularly visible in the range between the 800th and 1000th data point.

**Adaptive LASSA:** Here, the performance of the adaptive LASSA-based soft sensor is assessed. Figure 4.45 allows one to analyse different values of the three input parameters and the performance achieved by the soft sensors with corresponding parameter values.

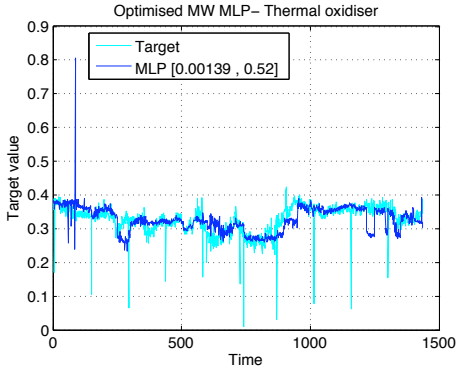
Figure 4.46 shows the achieved performance for the different parameter settings in form of a boxplot. It can be, similarly to the non-adaptive case, observed that the performance fluctuation remains very low. At the same the positions of the boxplot moves towards the lower MSE values. Interestingly, the median value does not change between the non-adaptive and adaptive case. The explanation of this can be found in Figure 4.45 where one can see that the performance of the models using $\sigma^{adapt} = 1.0 \cdot 10^{-2}$ is quite low, which keeps the median at a high value.



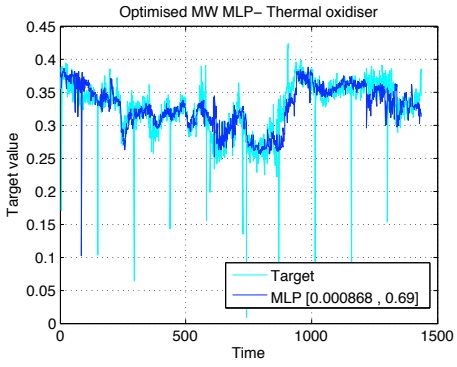

(a) Step size: 200 samples




(b) Step size: 100 samples




(c) Step size: 50 samples




(d) Step size: 10 samples

Figure 4.42: Thermal oxidiser: Effect of the adaptation of the PCA+MLP soft sensor using the moving window technique

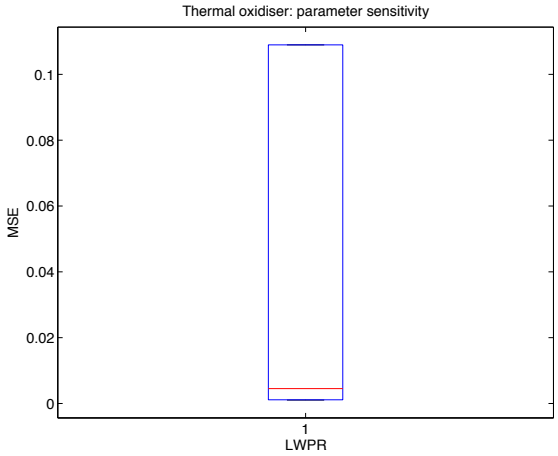



Figure 4.43: Thermal Oxidiser: Sensitivity of the LWPR method's performance with respect to its parameter settings

The settings that appear to be optimal for this data set are:

- $\sigma = 10^{-2}$

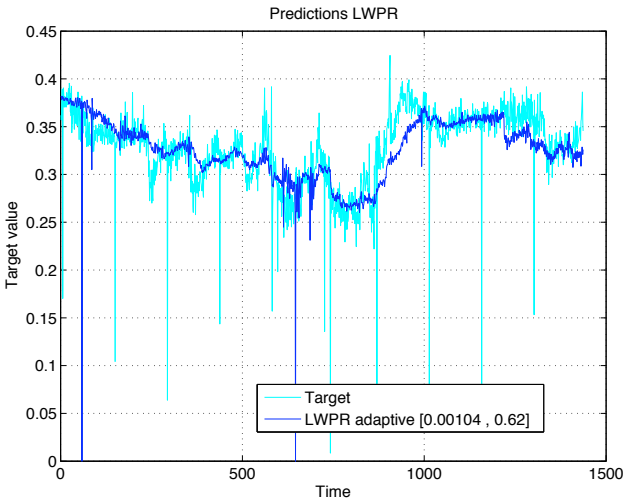



Figure 4.44: Thermal oxidiser: Predictions of the adaptive LWPR-based soft sensor

- $\sigma^{adapt} = 10^{-4}$

- $n^{init} = 50$

Figure 4.47 shows the predictions of the soft sensor based on the adaptive LASSA approach. In this case, a performance improvement compared to the non-adaptive version of this method can also be found (compare to Figure 4.39).

### Thermal oxidiser experiments summary

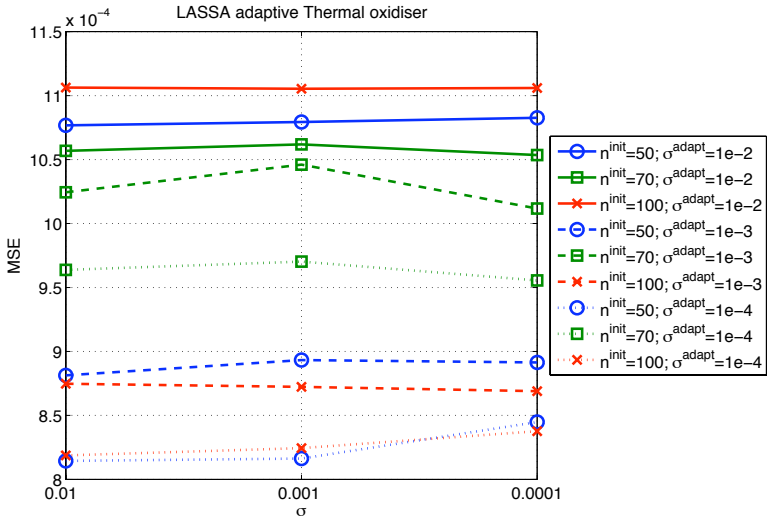
Analysing this data set, different patterns compared to the previous data set were observed. For this data set all of the applied adaptation methods led to performance improvement, demonstrating the benefit of applying the adaptation methods. This was best seen in the case of the PCA+MLP soft sensor combined with the moving window adaptation, where an increasing performance with decreasing step size, i.e. more frequent adaptation, was shown.

Another interesting characteristic is that the difference between the robust and conventional version of the PCA was not as large as for the industrial drier data, which can be accounted to lower number of outliers in this data set. Nevertheless, the robust version still achieves slightly better performance. The effect of model combination also had, like with the previous data, a strong influence on the performance and a reduction in the error variance and error level was observed.

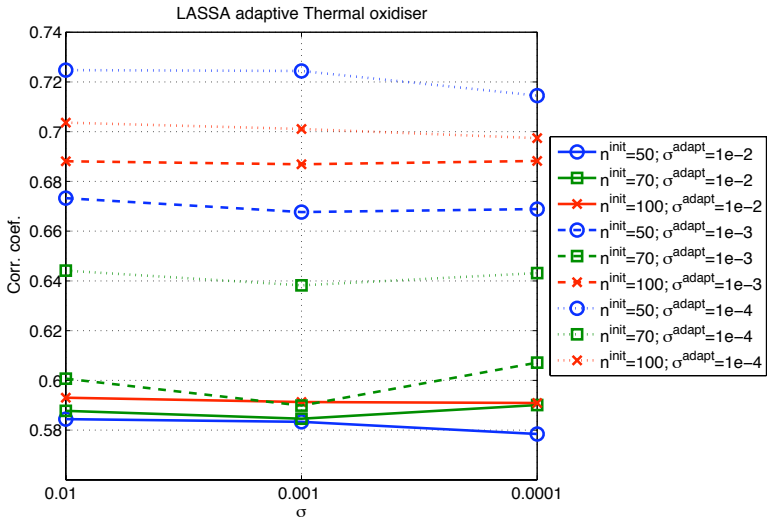
The LWPR-based soft sensors have shown to be very sensitive with respect to the parameter settings in both analysed scenarios.

In contrast to this, the LASSA-based soft sensor appear to be less sensitive to the parameter settings. Also the achieved error values were lowest from all of the analysed methods. This fact indicates that using LASSA is the best choice from the three algorithms for both soft sensor types.

The performances of all of the optimal models are summarised in Table 4.4.




(a) MSE




(b) Correlation coefficient

Figure 4.45: Thermal oxidiser: Influence of $n^{init}$, $\sigma$ and $\sigma^{adapt}$ on the performance of the adaptive LASSA-based soft sensor

### 4.4.6 Catalyst activation experiments

#### Non-adaptive soft sensors

**PCA+MLP:** Again, starting with the comparison between the conventional and the robust PCA shown in Figure 4.48 the dominance of the robust PCA is obvious. It can also be observed that the MSE values of the models reported in the figure are much higher than in the previous cases, indicating a poor performance of the models. In Figure 4.48(b) there is a missing point in the plot. For this point the correlation coefficient could not be calculated, supporting the weak performance hypothesis. The reason why the correlation could not be calculated is that one of the PCA+MLP

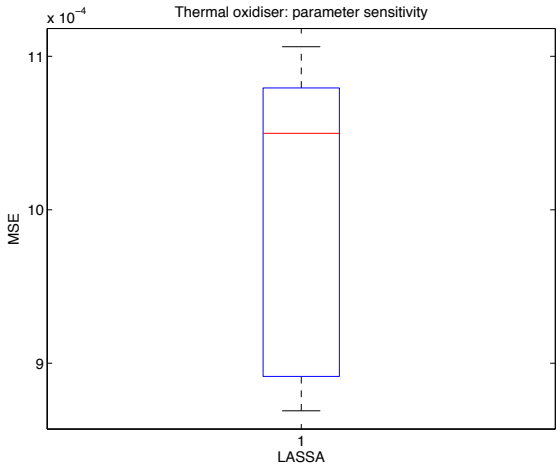



Figure 4.46: Thermal oxidiser: Sensitivity of the LASSA method's performance with respect to its parameter settings

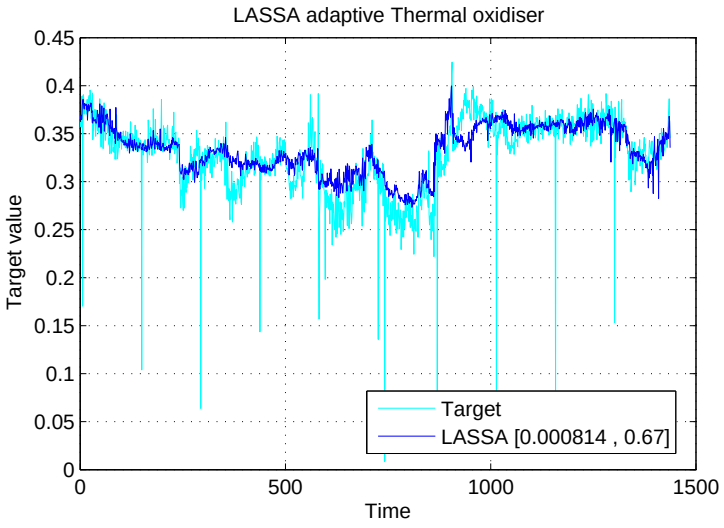



Figure 4.47: Thermal oxidiser: Predictions of the adaptive LASSA-based soft sensor

models predicts a constant value for all data instances, which leads to a division by zero during the correlation coefficient calculation.

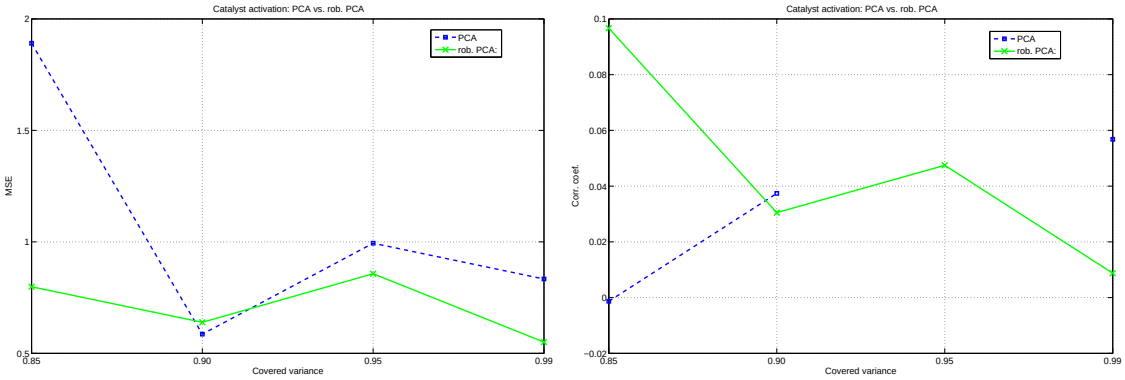
The analysis of the robust PCA in Figure 4.49 shows no obvious patterns, which makes the selection of optimal parameter value difficult. Nevertheless, the optimal variance coverage selected for further processing is $99\%$.

There are similar difficulties with the number of hidden units for the MLP. One can see that the achieved values stretch almost over the whole range of possible correlation coefficients, especially when considering the correlation coefficient figures (Figure 4.50(b) and 4.51(b)). As for the MSE figures, the same effects between the plain and combined models as for the other data sets can be observed, although in a milder form. Applying Occam's razor rule, the decision for the optimal number of hidden units is in favour of the simplest model-type, i.e. a single hidden unit.

In summary, the optimal PCA+MLP soft sensor for the catalyst activation data set has the following parameters:

| | non-adaptive | | | adaptive | | |
|---|---|---|---|---|---|---|
| | PCA+MLP | LWPR | LASSA | PCA+MLP+MW | LWPR | LASSA |
| MSE | $2.43 \cdot 10^{-2}$ | $2.17 \cdot 10^{-2}$ | $\mathbf{8.99 \cdot 10^{-3}}$ | $8.68 \cdot 10^{-3}$ | $1.04 \cdot 10^{-2}$ | $\mathbf{8.14 \cdot 10^{-3}}$ |
| Corr. Coeff. | 0.22 | 0.30 | **0.72** | **0.69** | 0.62 | 0.67 |

Table 4.4: Thermal oxidiser: Comparison of the MSE and correlation coefficient performances between the different soft sensing approaches




(a) MSE

(b) Correlation coefficient

Figure 4.48: Catalyst activation: Comparison between the performances achieved with PCA and robust PCA pre-processing

- Pre-processing: robust PCA covering $99\%$ of the variance
- Predictive technique: mean combination of ten MLPs with $numHid = 1$.

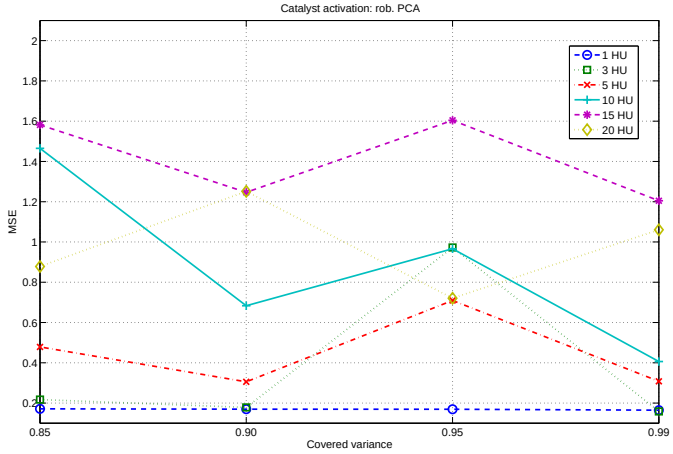
The predictions of such a soft sensor confirming the claims about the weak prediction performance can be found in Figure 4.52.

**Non-adaptive LWPR:** Figure 4.53 shows the range of MSE performances obtained by using different parameter settings. Again, it is possible to see that the performance is quite low. Nevertheless, the range of the achieved performances is significantly smaller than the one of the PCA+MLP based soft sensors shown in Figure 4.51.

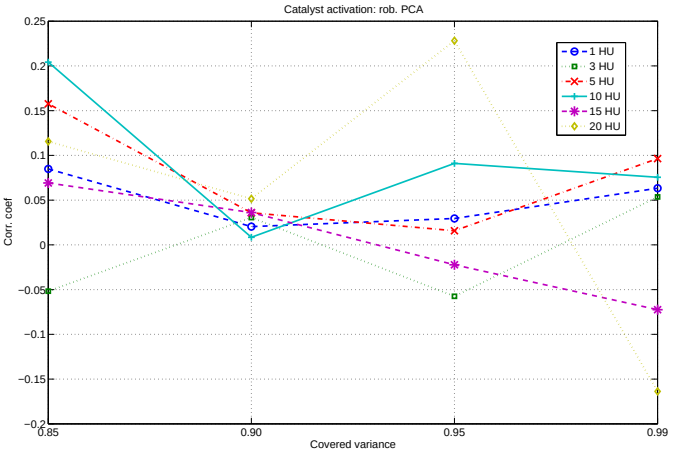
For the LWPR-based soft sensors, the optimisation on this data set led to the following parameters:

- $initD = 1$
- $wGen = 0.1$
- $penalty = 10^{-7}$

The predictions of the model are shown in Figure 4.54. For this data set, the non-adaptive LWPR-based model also fails to provide any useful prediction. In particular, the predictions show a strong offset and do not follow the trend of the target variable.




(a) MSE




(b) Correlation coefficient

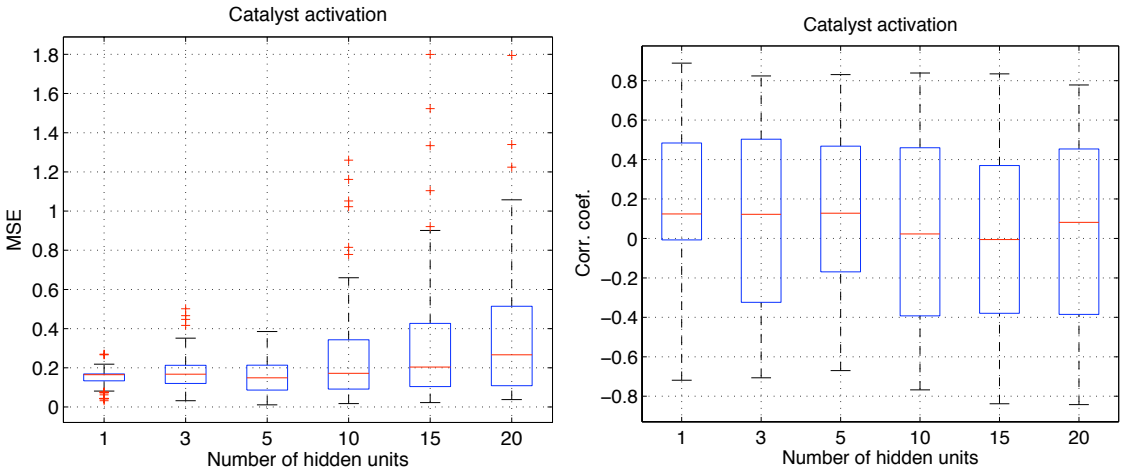
Figure 4.49: Catalyst activation: Influence of the variance covered by the robust PCA preprocessing and of the hidden units number on the performance PCA+MLP-based soft sensors

**Non-adaptive LASSA:** The first step of the LASSA-based soft sensor development is again the evaluation of the influence of $\sigma$ and $n^{init}$, which is presented in Figure 4.55.

When plotted altogether, as shown in Figure 4.56, it can be observed that the range covered by the boxplot is the best from all of the presented non-adaptive algorithms. Nevertheless, similarly to the other models, the performance level still remains very low.

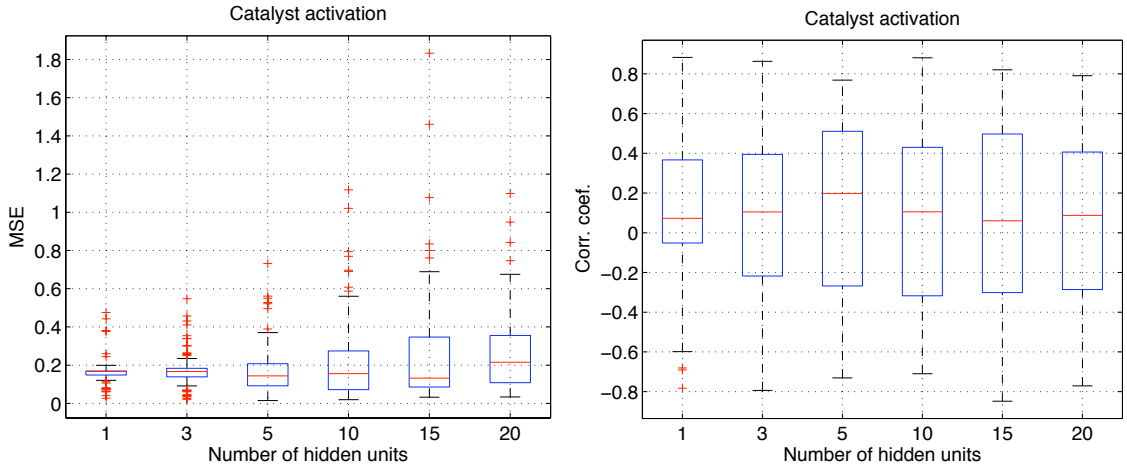
The optimal parameter values are: $\sigma = 10^{-4}$ and $n^{init} = 50$.

Using the above settings results in the construction of nine receptive fields for the training data. The resulting predictions of the non-adaptive LASSA-based soft sensor are shown in Figure 4.57. In harmony with the other non-adaptive approaches, the non-adaptive LASSA-based soft sensor fails to provide useful predictions for this data set, although the performance of this model was the best of all of the non-adaptive models.




(a) MSE (b) Correlation coefficient

Figure 4.50: Catalyst activation: Performance comparison between PCA+MLP-based soft sensors with varying number of hidden units




(a) MSE (b) Correlation coefficient

Figure 4.51: Catalyst activation: Performance comparison between combined PCA+MLP-based soft sensors with varying number of hidden units

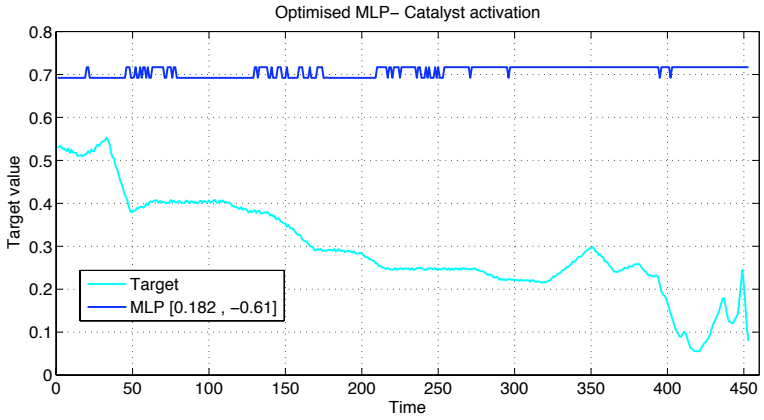



Figure 4.52: Catalyst activation: Predictions of the non-adaptive PCA+MLP-based soft sensor

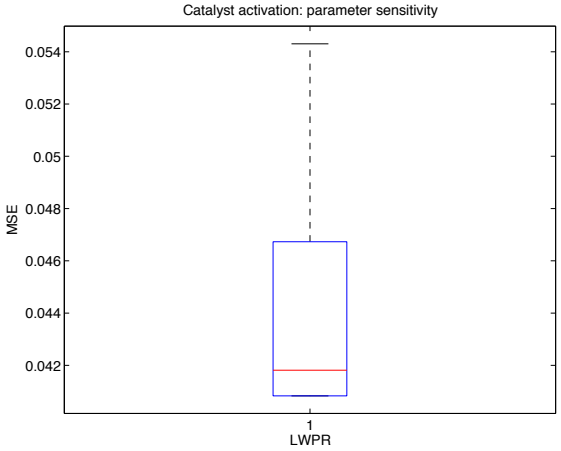



Figure 4.53: catalyst Activation: Sensitivity of the LWPR method's performance with respect to its parameter settings

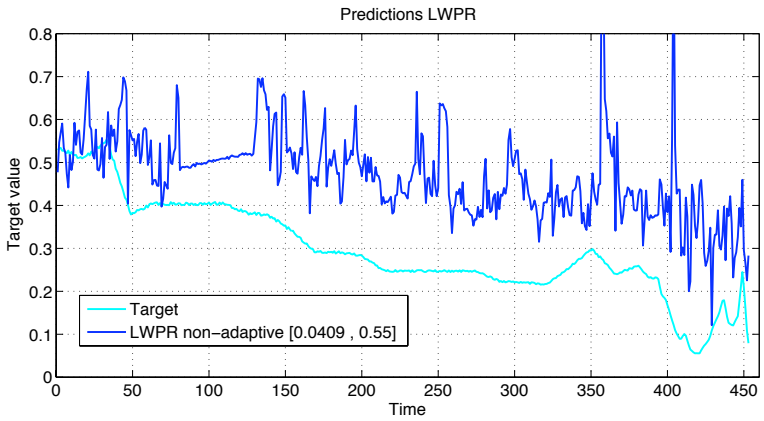
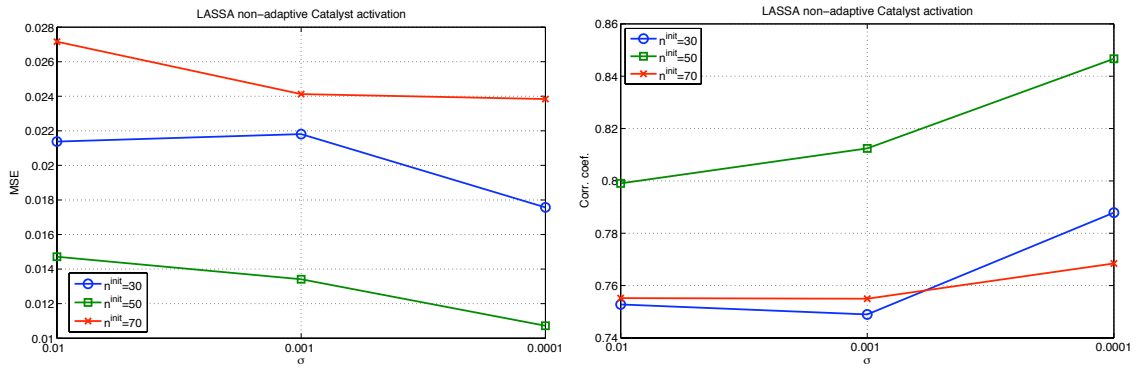



Figure 4.54: Catalyst activation: Predictions of the non-adaptive LWPR-based soft sensor




(a) MSE (b) Correlation coefficient

Figure 4.55: Catalyst activation: Influence of $n^{init}$ and $\sigma$ on the performance of the non-adaptive LASSA-based soft sensor

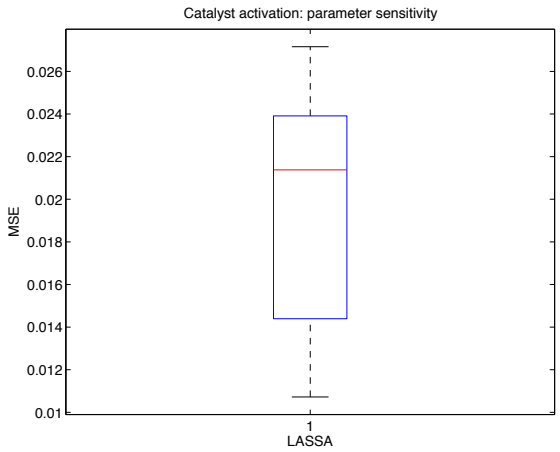



Figure 4.56: catalyst activation: Sensitivity of the LASSA method's performance with respect to its parameter settings

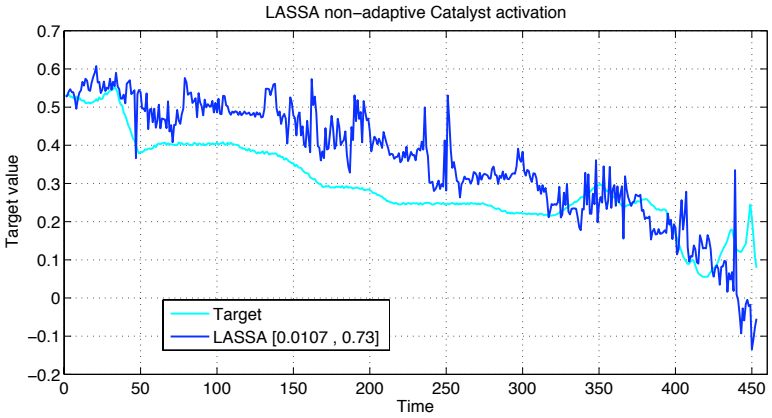
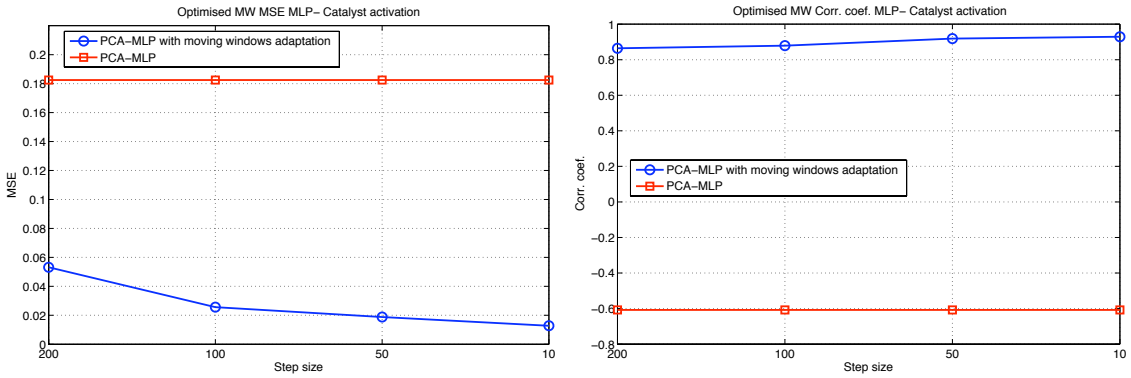



Figure 4.57: Catalyst activation: Predictions of the non-adaptive LASSA-based soft sensor

### Adaptive soft sensors

**PCA+MLP with moving window adaptation:** The influence of the retraining of the model using the moving window technique can be observed in Figure 4.58. The effect of the moving win-




(a) MSE (b) Correlation coefficient

Figure 4.58: Catalyst activation: Effect of the moving window step size on the performance of the adaptive PCA+MLP-based soft sensor

dow adaptation is obvious. There is a large improvement, especially for the correlation coefficient. Similarly to the thermal oxidiser data set, the performance is also increasing with the decreasing step size, indicating the need for frequent retraining of the soft sensor (see Figure 4.59). Although the predictions of the adaptive model are more accurate, there is still a large offset between the predicted and correct target values.

**Adaptive LWPR:** As expected, there is a significant performance improvement when applying the adaptive version of LWPR (see Figure 4.60).

The optimal parameters for the adaptive version of the LWPR soft sensor for this data set are:

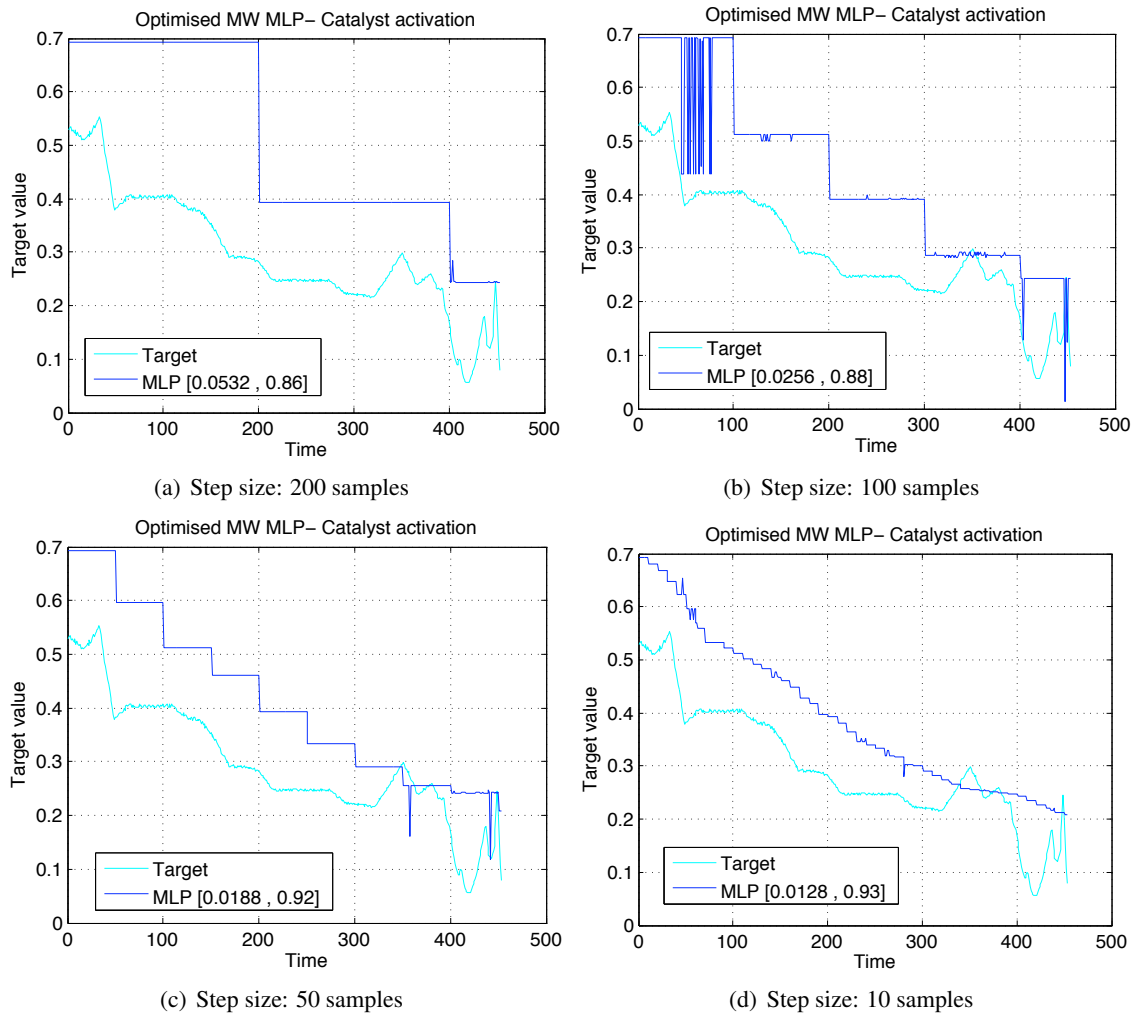
- $initD = 10$
- $wGen = 0.75$
- $penalty = 10^{-7}$.

The results achieved using this adaptive algorithm are reported in Figure 4.61. The predictions follow the trend, although there is still a slight offset and the adaptation is too slow to follow the dynamics of the target variable.

**Adaptive LASSA:** Figure 4.62 allows the analysis of the settings of the three input parameters of the adaptive LASSA-based soft sensors.

The settings that appear to be optimal for this data set are:

- $\sigma = 10^{-4}$
- $\sigma^{adapt} = 10^{-4}$
- $n^{init} = 50$.




(a) Step size: 200 samples

(b) Step size: 100 samples

(c) Step size: 50 samples

(d) Step size: 10 samples

Figure 4.59: Catalyst activation: Effect of the adaptation of the PCA+MLP soft sensor using the moving window technique

Figure 4.63 shows the predictions of the optimally parametrised soft sensor based on the adaptive LASSA approach. Contrary to the other adaptive approaches, there is no performance improvement in this case. This same can be observed when comparing Figure 4.64 with Figure 4.56. The reason for this is that the adaptation mechanism of the LASSA method does not involve the re-training or starting of new local experts. It merely incrementally updates the local expert descriptors $\mathcal{L}$ (see Equation (4.19)). This demonstrates the limits of the algorithm and motivates the application of more complex adaptation mechanisms.

### Catalyst activation experiments summary

For this data set, it was not possible to build well performing models based on the training data, which was demonstrated by the poor performance of all non-adaptive models.

Applying the adaptation mechanisms improved the performance. However, it was only the moving window and the LWPR techniques that led to the improvement. The reason for this is that these two techniques continuously re-train or adapt the predictors, which is not the case with

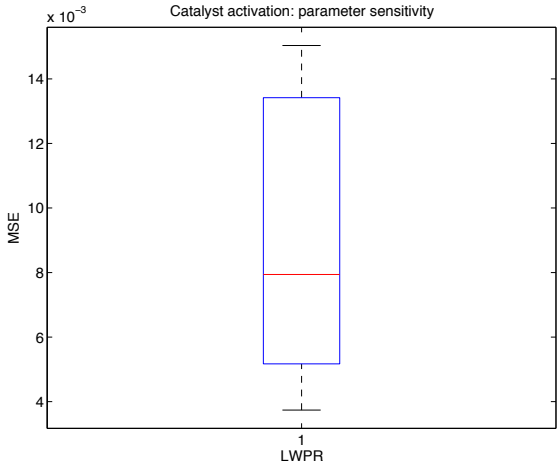



Figure 4.60: Catalyst Activation: Sensitivity of the LWPR method's performance with respect to its parameter settings

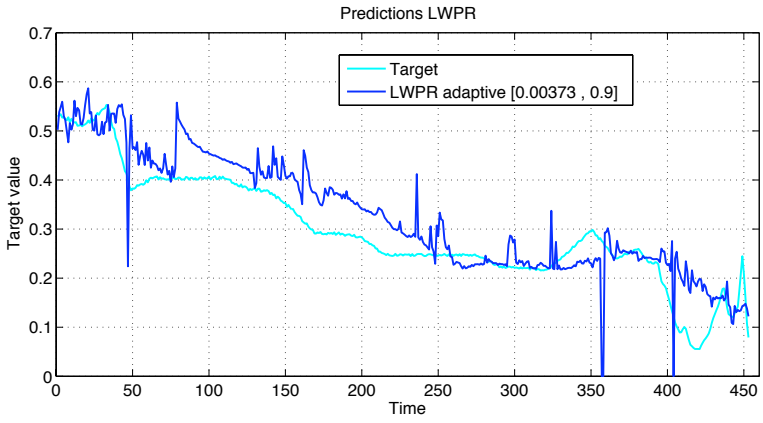
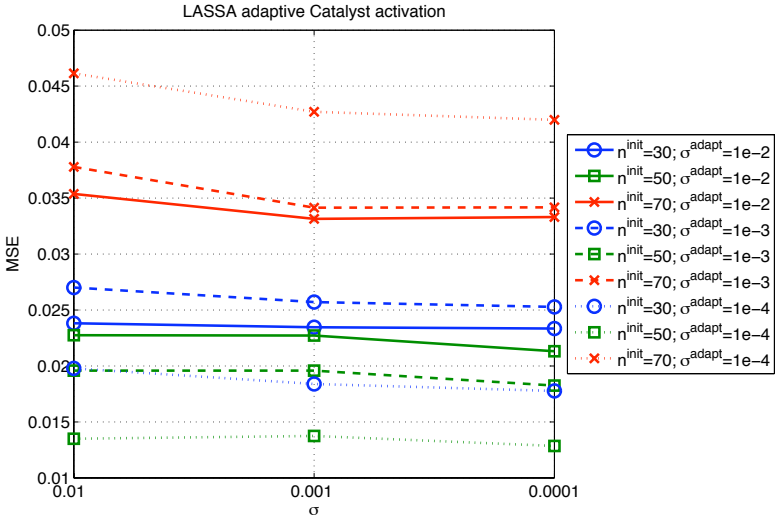


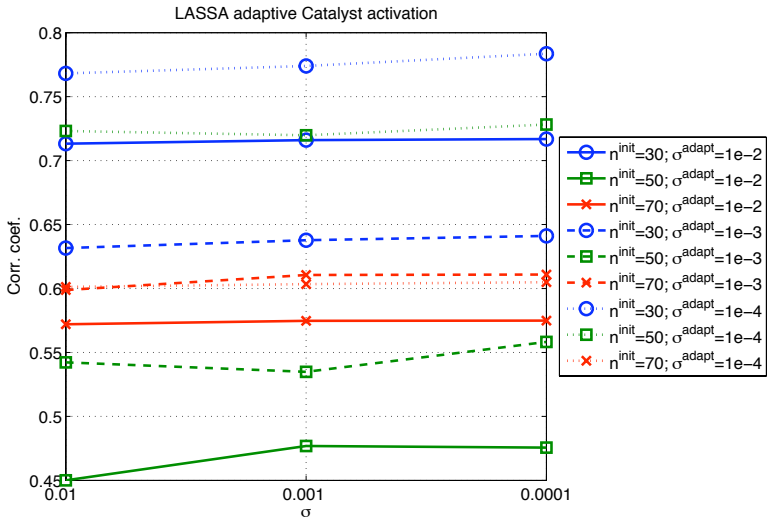

Figure 4.61: Catalyst activation: Predictions of the adaptive LWPR-based soft sensor

the LASSA-based soft sensors. The adaptation mechanism of the LASSA-based model is only adapting the combination weights whereas the local experts are not changed. This observation clearly demonstrates the limits of the LASSA method, which will targeted in the next chapter of this work.

The performances of all of the analysed soft sensor types are summarised in Table 4.5.




(a) MSE




(b) Correlation coefficient

Figure 4.62: Catalyst activation: Influence of $n^{init}$, $\sigma$ and $\sigma^{adapt}$ on the performance of the adaptive LASSA-based soft sensor

| | non-adaptive | | | adaptive | | |
|---|---|---|---|---|---|---|
| | PCA+MLP | LWPR | LASSA | PCA+MLP+MW | LWPR | LASSA |
| MSE | $1.82 \cdot 10^{-1}$ | $4.09 \cdot 10^{-2}$ | $\mathbf{1.07 \cdot 10^{-2}}$ | $1.28 \cdot 10^{-2}$ | $\mathbf{3.73 \cdot 10^{-3}}$ | $1.29 \cdot 10^{-2}$ |
| Corr. Coeff. | $-0.61$ | $0.55$ | $\mathbf{0.73}$ | $\mathbf{0.93}$ | $0.90$ | $0.85$ |

Table 4.5: Catalyst activation: Comparison of the MSE and correlation coefficient performances between the different soft sensing approaches

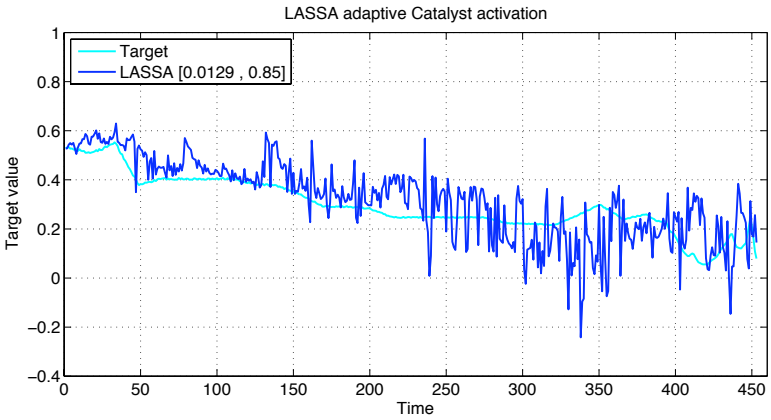



Figure 4.63: Catalyst activation: Predictions of the adaptive LASSA-based soft sensor

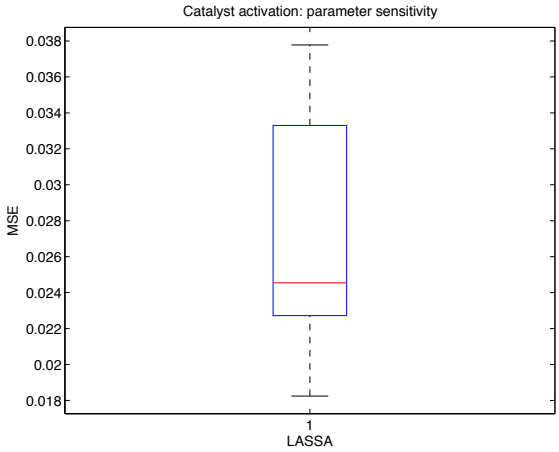



Figure 4.64: Catalyst activation: Sensitivity of the LASSA method's performance with respect to its parameter settings

## 4.5 Summary

There are various significant contributions in this chapter. The first one is a modified version of the state-of-the-art methodology for soft sensor development. In particular, the new methodology focuses on dealing with the adaptation aspects of soft sensing.

Based on the methodology, an adaptive local learning-based algorithm has been proposed in this chapter. A distinguishing characteristic of the algorithm is the novel data partitioning method that splits the data according to the validity of a simple prediction model. For each of the identified partitions of the data, a predictive model, called local expert, is built. The PCA pre-processing is also applied locally, which results in an efficient, locally focused, dimensionality reduction. Another characteristic of the algorithm is that it models the generalisation performance of the local experts in the locally reduced input-output space in the form of performance maps. These have two benefits, on one hand the performance maps can be used to obtain the combination weights for the local experts and on the other hand it allows the implementation of a simple adaptation mechanism. This adaptation mechanism does not require storing of any data as it only adapts the performance maps on sample-by-sample basis.

The next contribution of this chapter is a set of experiments applying (i) PCA+MLP: a state-of-the-art soft sensing approach; (ii) LWPR: a PLS-based incremental algorithm with particularly effective adaptation mechanism; and (iii) LASSA: the proposed soft sensing method to three industrial data sets. From the point of view of the data sets, it was shown that each of them requires a different degree of adaptation in order to develop useful models. The industrial drier data can be efficiently modelled with off-line methods, while the thermal oxidiser and the catalyst activation data sets require on-line adaptation in order to build well performing models.

In terms of data pre-processing, it was observed that there is a benefit from the application of the robust version of the PCA algorithm, which proves the capability of this method to deal with some of the issues of the process data.

From the point of view of the evaluated soft sensing methods, it has been shown that for the PCA+MLP approach the parameter selection can be difficult and even comprehensive selection does not necessarily lead to a well performing soft sensor. The moving window adaptation technique has also been shown to be difficult to deal with and if applied inappropriately it can also cause performance deterioration. Nonetheless, despite the difficulties a certain performance benefit from applying simple model combination methods was observed. The LASSA method proposed in this chapter has been shown to be easier to deal with indicating the usefulness of the local learning approach for soft sensing. In general the results have shown low sensitivity with respect to the parameter settings and the achieved error values were lower than those of the other methods. However, in the case of the catalyst activation data set, the adaptive version of the LASSA soft sensor failed to deal with the strong demand for adaptation. This is weakness of the method, which will be dealt with in the next step of this work. The effectiveness of the local learning approach was also demonstrated by the consistently well performing (especially in the adaptive case) LWPR-based soft sensors.

In summary, the empirical analysis has shown the potential of robust pre-processing, local learning and ensemble methods when aiming at the goals of this work. A conceptual framework for the development of robust and adaptive soft sensors, which reflects the findings of this chapter, is going to be proposed in the next chapter.

# Chapter 5

# Soft sensor development architecture

## 5.1 Introduction

This chapter presents a conceptual architecture that can be used for the development of adaptive predictive models. On the three step pathway introduced at the beginning of this thesis and shown in Figure 1.3, this architecture represents the first step.

Although focused on providing a concept for dealing with issues of industrial data sets, it can be used more generally for the development of any classification and regression models requiring robustness and adaptive capabilities.

The architecture defines a unified modular environment based on three concepts from machine learning, these are: (i) ensemble methods; (ii) local learning; and (iii) meta-learning, which are organised in a three layer hierarchy within the architecture.

For the actual predictions, which are performed at the lowest level of the hierarchy, any data-driven predictive method, such as an ANN, SVM, etc. can be implemented and plugged in.

Furthermore, particular attention is paid to its adaptation capabilities, which are discussed in Section 5.4. By exploiting its structure, there are several possibilities for the adaptive behaviour of the models developed according to the architecture. Arranged according to the overall three layer structure, these range from low level incremental, i.e. sample-by-sample, adaptation to more sophisticated adaptation techniques that result in on-line deployment of new models.

The structure and dependencies between the sections of this chapter are shown in Figure 5.1.

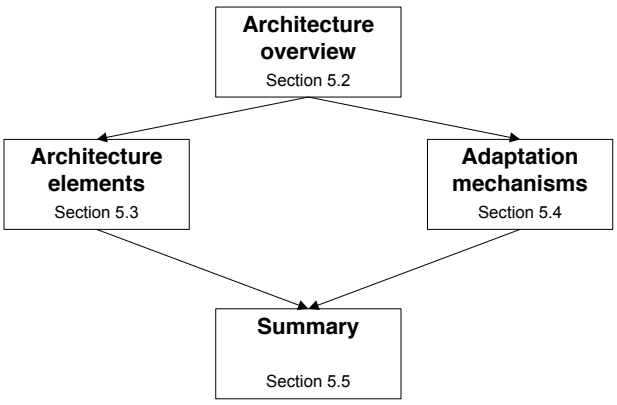



Figure 5.1: The structure of this chapter

## 5.2 Overview of the architecture

The main idea of the proposed architecture revolves around a certain degree of diversity represented by multiple competitive paths (predictive models) and their flexible combinations.

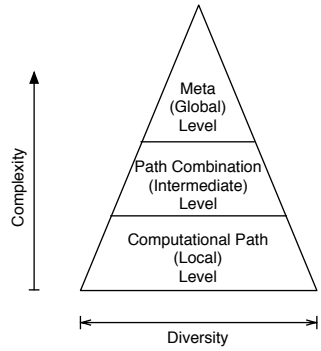



Figure 5.2: The three levels of information processing within the architecture

There are three hierarchical levels of information processing within the architecture. These are: (i) Computational Path (Local) Level; (ii) Path Combinations (Intermediate) Level and (iii) Meta (Global) Level. Figure 5.2 shows the hierarchy upon which the presented architecture is built. In the following, the three hierarchical levels are going to be briefly outlined.

**Computational path level:** As shown in Figure 5.2, this is the level with the lowest complexity but the largest diversity. This is achieved by maintaining a pool of diverse *computational paths*. In the terminology of this work, computational paths are basic information processing entities. Each computational path consists of none, one, or more pre-processing methods and one predictive technique, which maps the (pre-processed) input data onto the output space (see Figure 5.6 in the structure of the computational paths). Further details regarding the path structure and functionality can be found in Section 5.3.3. Special attention is also paid to the diversity among the paths as this plays an important role for the performance of the whole model. The role of diversity for ensemble methods was discussed in Section 3.4.3. The section also lists several diversity creation mechanisms including: (i) the initial conditions; (ii) the modelling method; and (iii) the training set structure. The proposed architecture allows the application of all of these diversity mechanisms at the level of the computational paths as it is going to be shown in Section 5.3.3. A particular mechanism for creating the training set structure diversity is local learning. Utilizing local learning methods also increases the flexibility of the architecture and allows the implementation of computational paths focusing on partitions of the input space, i.e. local experts. Using local learning is beneficial, especially in the case of industrial data where the information content of the available data is often limited and the underlying data structure is complex at the same time. Another positive effect of this approach is that data pre-processing, like feature selection, can be tuned locally, which is often more effective than applying global pre-processing. More details on local learning implementation within the architecture can be found in Section 5.3.5.

**Path combination level:** This level shows an increased level of complexity and decreased diversity. This is achieved by combining individual computational paths from the preceding level. At this level, the paths, which are competing against each other at the path level, are forced into

a collaborative behaviour. By doing this, teams of individuals that complement each other can be formed. Referring again to Figure 5.2, one can see that there is still a certain amount of diversity at this level as not only a single combination of predictors but a set of them is maintained. Managing a set of combinations again increases the flexibility of the architecture. Details of the path combination aspects in the architecture can be found in Section 5.3.4.

**Meta level:** At the top of the complexity pyramid is the meta level. From this level the whole model is controlled to optimise the predictions in terms of the global performance function, which is the actual function that has to be optimised. It can be approached by (i) controlling the populations at the lower levels, e.g. by launching paths to cover unexplored parts of the input space; (ii) looking for relations between parameters of the paths and the achieved performance; (iii) adapting the combinations, i.e. teams of experts, in order to reflect the current state of the data. Another task performed at this level, which corresponds to the traditional understanding of meta-learning, is the extraction, storage and transfer of knowledge across the different parts of the model as well as building the high-level knowledge of the architecture. In summary, using meta-learning capabilities facilitates the usage of high-level mechanisms for learning and building the high-level knowledge of the architecture. Furthermore, the meta-learning can take control of the parameters of the methods at the lower levels of complexity and adjust these according to the global strategy defined by the user. In this way, an abstraction layer between the model operator and the lower layers of the architecture is built. The meta-learning mechanisms within the architecture are explained in further detail in Section 5.3.6.

**A schematic overview:** A schema of the interactions within the architecture can be found in Figure 5.3. Due to the focus on data-driven modelling, the model is built upon the data. The data is the basis for the training and the evolution of the paths and path combinations. The meta level methods also draw their knowledge from the underlying data and control the lower levels (i.e. path and path combination levels) of the architecture in order to achieve the globally set strategy. The expert knowledge provides means to influence the behaviour of the architecture at all three levels of complexity. If required, this functional capability can be exploited for the intervention by the model developer in order to change the behaviour of the architecture or, as it was already mentioned, to alter the global goals. Examples of such intervention are selection and parametrisation of the pre-processing methods at the path level or manual selection of models to be combined at the path combination level. Furthermore, the possibility to implement the first principle (i.e. phenomenological) models at the path level is provided in this way.

Table 5.1 gives a summary of the main concepts represented within the architecture and of their main goals.

| **Technique** | **Purpose of introduction** |
|---|---|
| Concept drift detection and handling | Dealing with changing environment, keeping validity of the model |
| Ensemble building and diversity management | Improving prediction performance, providing different prediction and adaptation mechanisms |
| Local learning | Dealing with sparse data, limiting the complexity of the applied models |
| Meta-learning | Implementing global strategy, building abstraction layer between user and model |

Table 5.1: Main concepts represented within the architecture

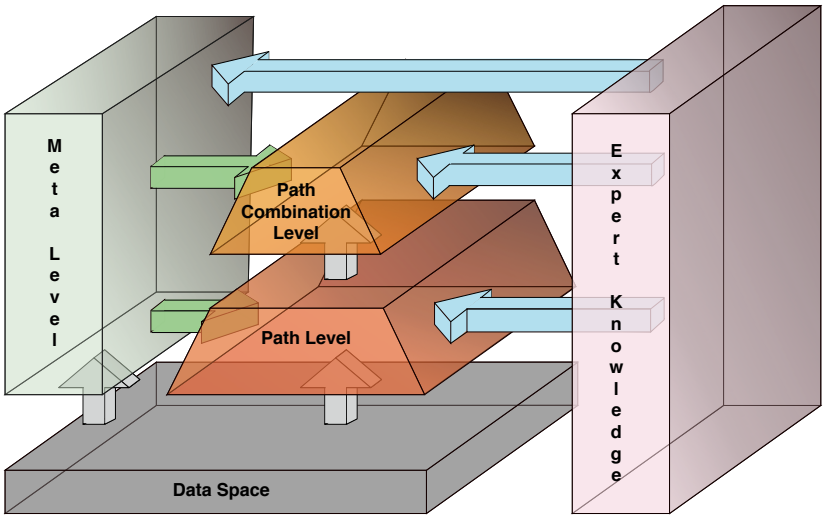



Figure 5.3: Overview of the interactions between the modules of the architecture

### 5.2.1 Adaptation capability of the architecture

The adaptation capability follows the hierarchy of the architecture and is present across the three different levels of model complexity. The adaptation loops and the interaction between the three levels of complexity is schematically shown in Figure 5.4. The figure shows the self-adaptation capability of the local and intermediate levels (see loops $a$,$b$ in Figure 5.4). In contrast to this, from the meta-level there is a connection to the lower levels (loops $c$,$d$). The particular mechanisms for

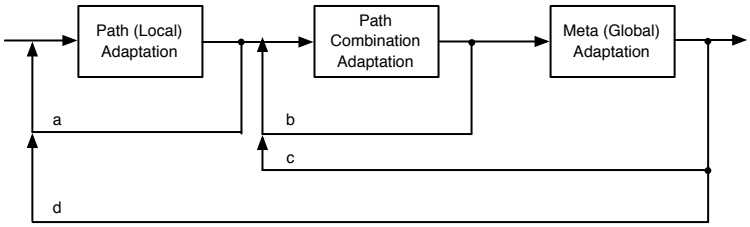



Figure 5.4: Adaptation loops available within the architecture

adaptation can be, for example, approaches dealing with concept drift discussed in Chapter 3. The adaptation functionality of the architecture is addressed in further detail in Section 5.4.

## 5.3 Elements of the architecture

This section describes the elements represented within the particular modules of the architecture in detail. Figure 5.5 presents a detailed view of the architecture with all the main modules including:

- Two pools of methods (PPMP, CLMP)
- Data Source module
- Paths module

- Path Combinations module
- Instance Selection Management module
- Meta-Level Learning module
- Global Performance Evaluation module

Furthermore, there are four connection types:

- Data Stream connection, which distributes the data to the remaining modules of the architecture.
- Evaluation Results, which is another data object carrying the results of evaluation of the predictions.
- Control Connections, transporting control information originating from the Instance Selection, the Meta-Level Learning or from the Expert Knowledge module.
- Pooling Connections, indicating which of the methods from the pools are used within the Paths or Path Combinations module.

As one can see, the architecture is rather general and provides a lot of degrees of freedom for its implementation, which in turn provides the possibility to focus on a particular functionality required by the task to be solved.

### 5.3.1 Data source

This module acts as an interface between the physical database or any other type of data storage and the actual architecture. The data are encapsulated into special data objects, $\mathcal{D}$, which are distributed across the architecture. These objects consist not only of the input data $X$ and target data $Y$ but also of basic statistical information $S$ (e.g. variable distributions), which can be useful for the other parts of the architecture. The data object has the following form: $\mathcal{D} := \{(X, Y), S\}$.

Additionally, there are different regimes of the data provision possible. For the traditional learning scenario, the data is distributed in batches of training, validation and test data. In a scenario dealing with an industrial modelling task there can be a set of historical data distributed to the other modules during the training phase in batch mode followed by a stream of single data instances during the on-line phase (see Section 2.3.1 for details of data provision in industrial modelling).

The way in which the data is distributed to the model is controlled using the connection $c^{e \to d}$.

### 5.3.2 Method pools

The next part of the architecture includes two pools of methods, namely the Pre-Processing Methods Pool (PPMP) and the Computational Learning Methods Pool (CLMP). These pools are repositories of objects that represent the data pre-processing and computational learning techniques instantiated and used within other parts of the architecture. The role of the pools is limited to providing the incorporated objects (e.g. data normalisation in PPMP or Multi-Layer Perceptron in CLMP) to the other parts of the architecture.

Communication with other parts of the architecture is carried out using *pooling connections* $p$. These connections indicate that a certain computational path is using an instance of the object (e.g.

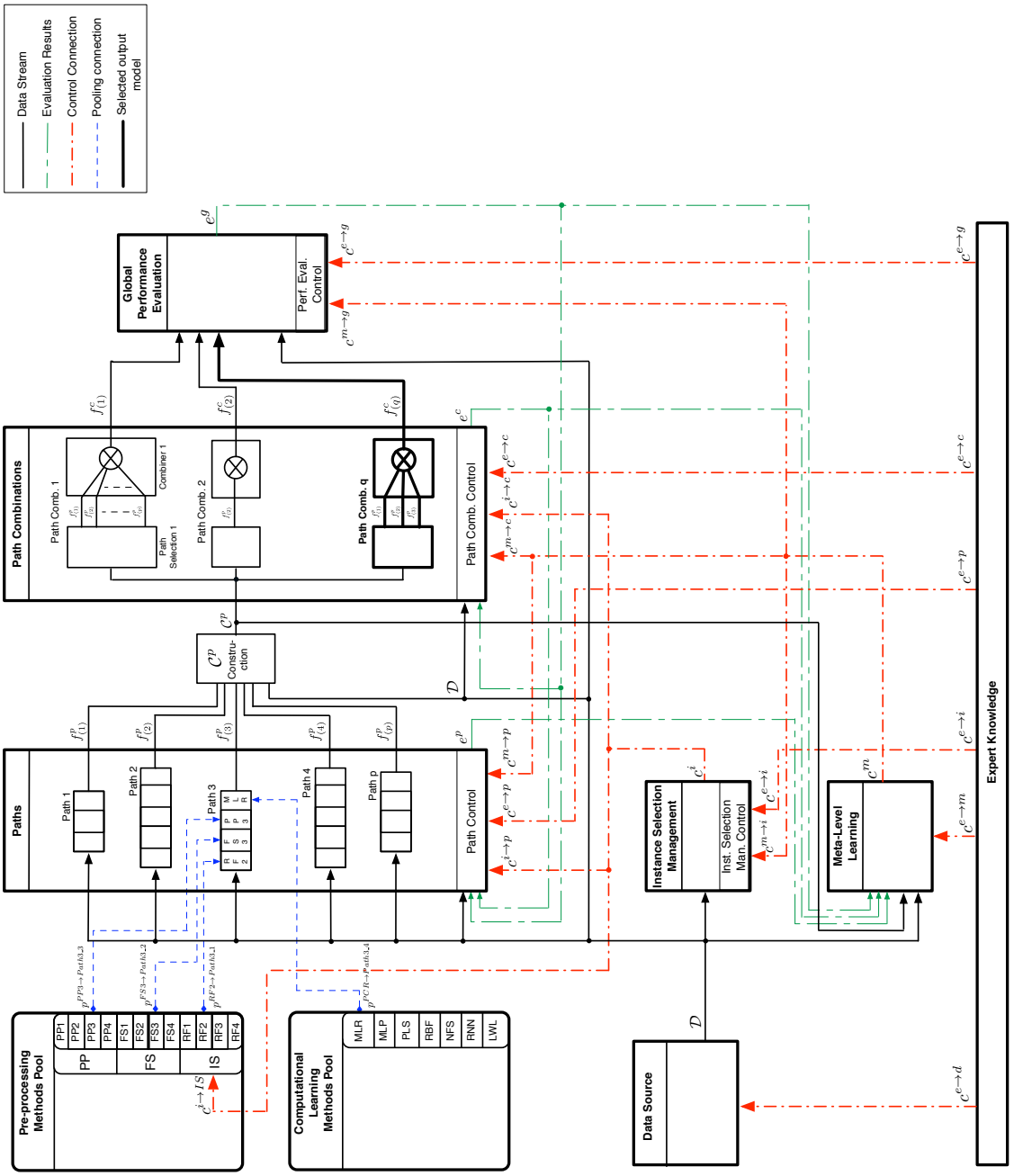



Figure 5.5: General overview of the proposed architecture for the development of robust and adaptive soft sensing algorithms

$p^{PP3 \rightarrow Path3\_3}$, which indicates that $Path3$ uses the pre-processing method $PP3$ from PPMP and places it at the third position within the path). Figure 5.5 shows an example where Path3 uses objects RF2, FS3 and PP3 from PPMP.

The pre-processing methods pool is further split into three different sub-pools: (i) Pre-Processing (PP) methods (outlier detection, normalisation, etc.); (ii) Feature Selection (FS) methods (correlation-based feature selection, etc.); and (iii) Instance Selection (IS) methods containing instance filters in the form of Receptive Fields (RF) used for the local learning functionality.

### 5.3.3 Computational Path

This module implements the lowest level of the hierarchy. The task of this module is to maintain a competitive environment where the CPs are managed. The management involves the building, adapting and removing of the particular paths, which are all controlled from the Path Control of the module.

The input of this module is formed by the following signals:

- $\mathcal{D}$: Data object from the Data Source
- $e^c$: Performance (e.g. Mean Squared Error) of the path combination objects managed in the Path Combinations module (see Section 5.3.4)
- $e^g$: Information about the performance of the whole model at the global level coming from the Global Performance Evaluation module (see Section 5.3.7)
- $c^{i \to p}$: Control information from the Instance Selection Management, for example notifications about the appearance of a new receptive field that triggers the deployment of a new computational path dealing with the new receptive field
- $c^{m \to p}$: Meta-level control information from the Meta-Level Learning module (see Section 5.3.6), for example, the adaptation rate for a particular computational path
- $c^{e \to p}$: This control signal provides a possibility to involve expert knowledge into the decisions made in the Path Control (see Section 5.3.8), this is, for example, an explicit use of a pre-processing method for all paths

Given all this information, the Path Control can implement various strategies for the launching, adaptation and removal of computational paths.

The actual computational paths consist of several elements, as shown in Figure 5.6. Among them there can be one or more pre-processing steps from the PPMP and one computational learning method from the CLMP. Furthermore, to be able to assess the local (path-level) performance, there is a need for local evaluation provided by the Local (Path) Evaluation unit that calculates the performance of the path $e^p$ according to the locally[1] defined performance measure. This is set and controlled using the $c^p$ control signal from the Path Control. The performance data is then provided to the Local Control Unit (LCU), which sends the control information $c^{lp}$ to the path elements and to the Local (Path) Memory. This control information can be, for example, a trigger for the adaptation of the path elements. The Local Control Unit receives not only the path performance data, but also information from the higher levels (i.e. from Meta-Level Learning through Path Control) of the architecture, which can additionally influence the adaptation steps. The Local (Path) Memory is responsible for storing past data instances and/or information extracted from them (e.g. variable distribution and statistics). The stored information plays an important role for the adaptation of the computational path (for details see Section 5.4).

The embedding of the computational path presented in Figure 5.6 into the Paths module is shown in Figure 5.12.

The computational path can be operated in three different modes with flexible switching between them performed from the LCU. These are:

[1] In this context local refers to the particular computational path rather than local in terms of the data space as used in local learning.

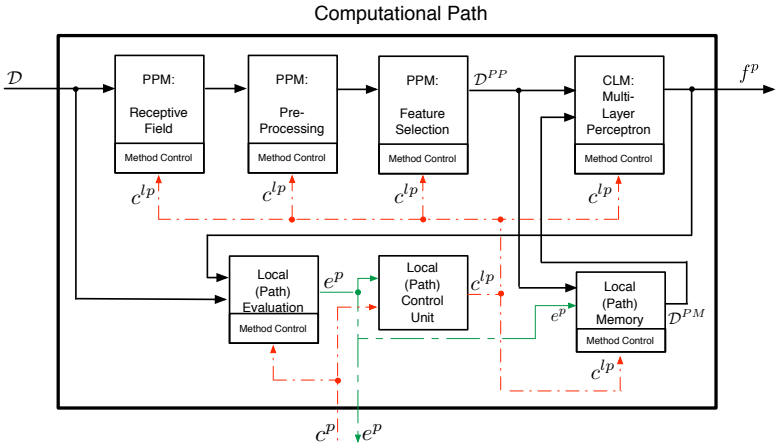



Figure 5.6: Computational path structure

**Training mode:** In this mode the path is trained. This applies to the Computational Learning Method as well as to the pre-processing methods within the path (e.g. training of PCA as the dimensionality reduction method). The set-up of the computational path during the training phase is presented in Figure 5.7.

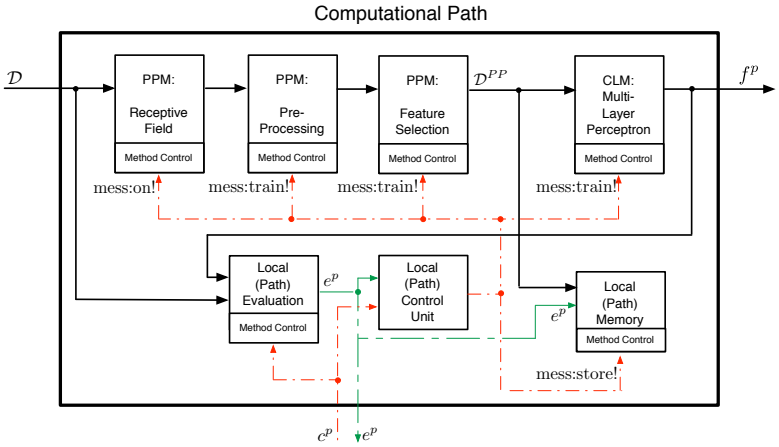



Figure 5.7: Computational path in the training mode

**Prediction mode:** For the prediction mode where the path is required to provide predictions of the target values given the input data instances, the Receptive Field element of the path can be deactivated since the path is requested to provide predictions to all data samples independent of their receptive field origin. Depending on the scenario and the adaptation mechanism, the data instances during this phase may also be stored in the Local (Path) Memory. An example of a computational path in prediction mode is shown in Figure 5.8.

**Incremental mode:** In this mode the data arrives sequentially either in the form of single instances or groups of them. The task of the CP is to implement the adaptation strategy at the lowest level (see Figure 5.4 - loop $a$). Once a new data instance arrives, the path makes a prediction of the

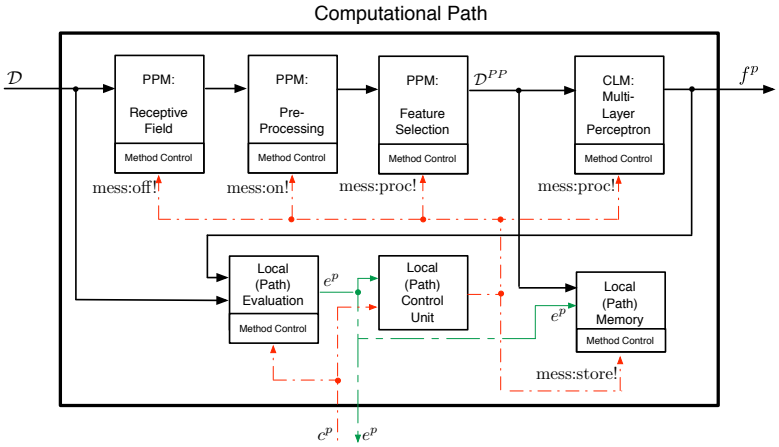



Figure 5.8: Computational path in the prediction mode

target value. When the correct target value becomes available, the performance of the predictions is assessed in the Local Evaluation and passed to the Local Control Unit, which makes a decision about whether the path should be adapted or not. In the case when a decision in favour of the adaptation is made, the particular elements of the path are adapted using the implemented adaptation strategy controlled from the Local Control Unit. For example, in the case of the moving window adaptation, the samples stored in the Local Memory are retrieved and passed to the learning method, which uses these data instances for retraining of the path elements (both pre-processing as well as the prediction methods), as shown in Figure 5.9.

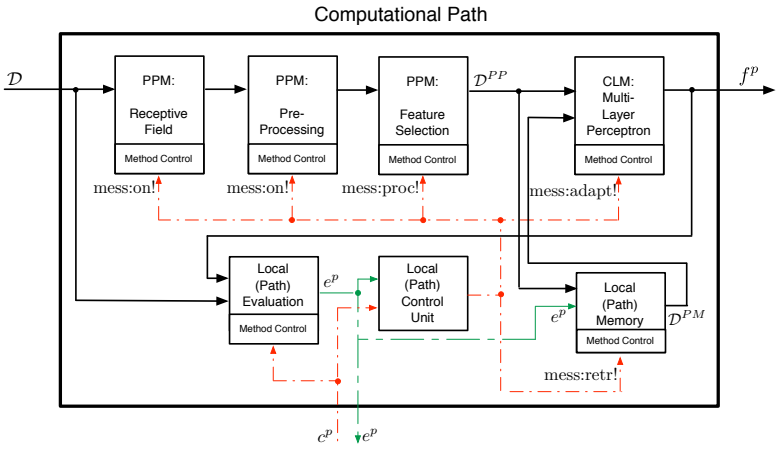



Figure 5.9: Computational path in the incremental mode

### 5.3.4 Path Combinations

For the reasons discussed in Section 3.4 there is a special module, namely the Path Combination module, devoted to ensemble building techniques. This module operates at the intermediate level of complexity pyramid shown in Figure 5.2. At this level, the individual paths that are competing against each other at the path level are merged to groups where they cooperatively perform the target value prediction.

In the traditional combination scenario there is a set of models whose predictions are combined by a combiner. In order to increase the flexibility of the architecture, it goes further and provides a possibility to manage a set of independent model combinations or even their hierarchies (as shown, for example, in [155, 149]).

The particular elements within the Path Combinations module are managed from the Path Combination Control (PCC) of the module.

Figure 5.5 shows that the predictions of particular computational paths $f^p(X)$, together with the target values $\mathbf{y}$ from the data object $\mathcal{D}$, are aggregated to a new data object $\mathcal{C}^p := \left\{(f^p_{(k)}(X), \mathbf{y})\right\}_{k=1}^{p}$. This data object describes a new data space, which is formed by the paths predictions and the target value. The aim of the Path Combinations (PC) is to make a prediction of the target value in this space. In a simple case, it can be a linear combination of the input variables: $f^c = c(\mathcal{F}) = \sum_k w_k f^p_{(k)}$ or, in a more complex case a non-linear transformation performed by a model like Multi-Layer Perceptron for instance. The advantage of such a representation, which is very similar to the one at the lower (i.e. path) level, is that the combinations can be represented in the same way as the computational paths only operating in the space formed by the data object $\mathcal{C}^p$ (compare Figure 5.6 with Figure 5.10). In particular, it means that one can apply feature selection methods to select a subset of paths (i.e. s subspace of $\mathcal{C}^p$) that have to be combined.

Furthermore, the Path Combinations can be operated in the same three modes, i.e. training, prediction and incremental learning, as it was the case for the computational path, and the same approaches can be used for the implementation of three modes (e.g. moving window adaptation for the incremental mode).

The embedding of the combinations into the Path Combinations module of the architecture is also shown in Figure 5.12.

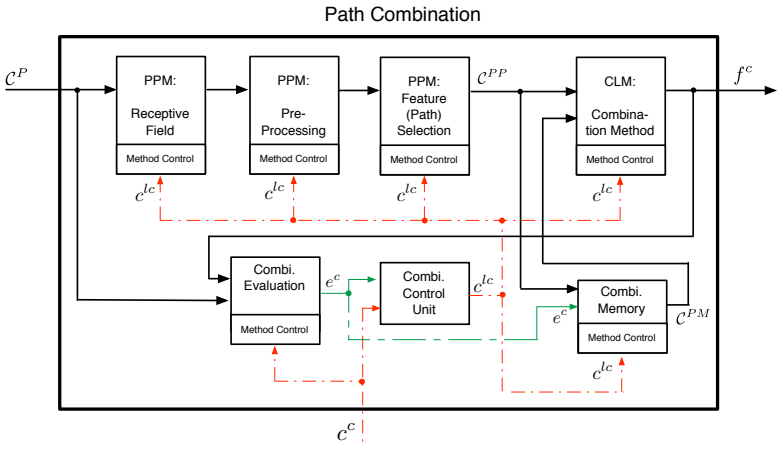



Figure 5.10: Path combination structure

### 5.3.5 Instance Selection Management

The Instance Selection Management (ISM) module is responsible for the partitioning of the input data space into locally coherent sub-spaces called *Receptive Fields* (RF). Having this functionality, the architecture can incorporate local learning models.

The construction of the receptive fields can for example be done using clustering algorithms (e.g. k-means [123]) or any other data partitioning method such as the algorithm presented in

Section 4.3.1.

The ISM module is managed from the Instance Selection Management Control, which has the following two signals as input:

- $c^{m \to i}$: Control connection from the Meta-Level Learning module, which can provide parameters like the number of receptive fields to be managed or threshold parameters for the deployment of new receptive fields.
- $c^{e \to i}$: Expert knowledge, which can overwrite the $c^{m \to i}$ signal or manually define receptive fields that are of advantage for the global model from the expert's point of view.

Providing the full local learning functionality involves several parts of the architecture. The ISM handles the deploying of new receptive fields, their adaptation and removing. Furthermore, the ISM provides the information about the receptive fields to the Pre-Processing Methods Pool using the control connection $c^i$.

According to the three globally defined modes of operation, the ISM module can be run in the following modes:

**Training mode:** In this mode the ISM receives a batch of historical data and its task is to split the data into local partitions. The number of the partitions may be provided by the Expert Knowledge (i.e. by the user) if available. If not, techniques for the estimation of the number of receptive fields, like parameter cross-validation in MLL or techniques that derive the number of local partitions automatically like the algorithm presented later on in Section 6.2.1 have to be applied.

**Prediction mode:** In this mode there is no task for the ISM and it can simply be deactivated or it can be switched to the incremental mode (see below) in order to adapt the receptive fields.

**Incremental mode:** As the goal of the incremental mode is to make predictions of the incoming data instances and to adapt the model at the same time, the task of the ISM is to update the receptive fields depending on the on-line data. This task includes deploying new receptive fields, adaptation and the removing (pruning) of the existing ones. Examples of such approaches were presented in [33, 61, 62, 207].

### 5.3.6 Meta-Level Learning

The Meta-Level Learning (MLL) module of the architecture is responsible for high-level learning, control and decision making. On the basis of the collected information from the other parts of the architecture, a picture of the global behaviour of the architecture is constructed in this module.

The input to the MLL module is formed by:

- $\mathcal{D}$: The data object
- $\mathcal{F}^p$: The paths predictions object
- The evaluation results signals from all levels across the architecture, which are:
  - $e^p$: The evaluation signal of the computational paths
  - $e^c$: The performance evaluation of the path combinations
  - $e^g$: The evaluation results from the Global Performance Evaluation module

- $c^{e \to i}$: The control connection coming from the Expert Knowledge module. This signal is especially useful, for example, for setting the parameters of the applied meta-level technique.

The role of this module can be to learn the dependency between the methods from the pools and the performance at the different levels of complexity, which are collected across different receptive fields, datasets, initial parameter sets, etc. There are several approaches dealing with this task that can be utilised at this level. Examples of these are:

- Statistical/Information theoretical meta-attributes [99]
- Model based meta-attributes [10]
- Landmarking [141]

### 5.3.7 Global Performance Evaluation

Referring again to the traditional modelling scenario, the prediction performance of the developed model is usually based on performance measures like the Mean Squared Error (MSE), which is universally applied when checking the prediction performance of a model.

The Global Performance Evaluation (GPE) module goes further and provides a flexible way to implement various performance measures. This does not necessarily need to refer to the predictive performance only. Several performance measures, which may be considered in the architecture, are discussed in [67]. Examples of such methods are: compactness, computational complexity, etc.

This module can also incorporate multi-objective evaluation techniques. Extending the previous example the performance measure could be a combination of the models' performance prediction and their diversity.

The output of the GPE module plays an important role for the whole model as it is used in the decision making at all three levels of information processing (see signal $e^g$ being input to the Paths, Path Combinations and Meta-Level Learning modules). For this reason, there is a default performance measure implemented (e.g. the MSE). However for cases where another performance measure is required, there is mechanism for changing the default measure using the Expert Knowledge module and the control signal $c^{e \to g}$ available. Another task performed in the GPE module is the selection of the output model, i.e. the local expert or path combination providing the final predictions. This can be controlled either automatically from MLL or manually by the user (through the Expert Knowledge module) using the signals $c^{m \to g}$ or $c^{e \to g}$ respectively.

The GPE module also acts as the output module, which presents the predictions results and other information about the model to the user.

### 5.3.8 Expert Knowledge

This part of the architecture provides an interface for the interaction between the model operator and the model. Using this interface, the operator gets access to the particular modules and can manually influence the operation of the parts of the architecture. For the high-level parts of the architecture (Meta-Level Learning and Global Performance Evaluation), this module is used to define the techniques operating at this level and to set their parameters. In the case of the Global Performance Evaluation the expert has the possibility of setting the performance evaluation functions if the default measure is not desired.

The particular control connections from the Expert Knowledge module and possible examples of information that can be distributed using these connections are:

- $c^{e \to p}$: In this case the expert can manually control either the computational paths themselves, their parameters or the Path Control where decisions about the adaptation, launching and removing of the paths are made. In a particular example the expert can, based on the available knowledge about the data, define necessary pre-processing steps and their parameters (e.g. PCA with ten principal components).

- $c^{e \to c}$: Like the previous case, the expert can influence either the decision making within the Path Combination Control or suggest some path combinations (e.g. combine paths 1, 3 and 5 using the mean ensemble building method) that have to be included in the combinations population.

- $c^{e \to g}$: The task of this signal is the definition of the global performance evaluation function, for example balancing the MSE and correlation coefficient values of the predictions as a multi-objective performance measure.

- $c^{e \to i}$: This signal provides the expert the possibility of influencing the building of new receptive fields and the adaptation and removal of available ones. In a particular case, the signal could carry the numbers of instances that, according to the expert's knowledge, represent a particular state of the data and have to be presented as a separate receptive field.

- $c^{e \to m}$: This connection carries the information about the method that has to be applied at the meta level and its parameters. Relating to the examples mentioned in Section 5.3.6, it could be the type of meta-features and meta-learner that have to be applied in the MLL module.

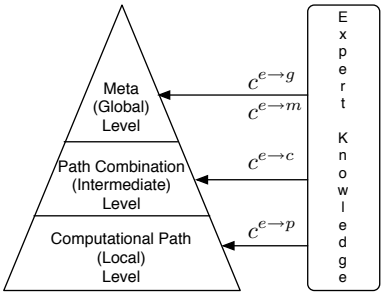



Figure 5.11: The interaction between the Expert Knowledge module and the architecture's three levels of information processing

Figure 5.11 shows the hierarchical structure of the architecture and the interaction possibilities of the Expert Knowledge module at the different levels of the pyramid.

Another task controlled from the Expert Knowledge module can be the switching between the three operating modes of the model. From here, the control signals $c^e$ are used for switching of the Paths, Path Combinations, Instance Selection Management and Meta-Level Learning modules between the training, prediction and incremental modes.

## 5.4 Adaptation mechanisms

One of the key features of the proposed architecture is its ability to deal with dynamic environment represented by changing data. In order to use the adaptation capabilities of the architecture the particular modules have to be switched to the incremental mode. In harmony with the three levels of complexity, there are also three different levels of adaptation possible, namely adaptation at:

- Path level
- Path combination level
- Meta level.

Figure 5.12 shows in a detailed way the adaptation mechanisms and their interaction possibilities and the next three paragraphs provide a detailed discussion of the adaptation mechanisms at the three different levels.

### 5.4.1 Path level adaptation

The adaptation at the path level refers to the adaptation ability of the particular computational paths. At this level, the paths can be adapted in two different ways. On one hand using the knowledge about their own performance, which is also referred to as the *self-adaptation*, (see loop $a$ in Figure 5.4 and $e_1^p$ in Figure 5.12) or on the other hand, by using the global control coming from the higher level of the architecture (represented by loop *d* in Figure 5.4 and connection $c_1^p$ in Figure 5.12).

The self-adaptation loop consists of the feedback of the prediction, which is compared to the correct target values in the Local (Path) Evaluation unit (see Figure 5.6). Given the implemented local error measure, the error $e^p$ between the prediction $f^p(X)$ and the correct value $y$ is calculated in the Local Evaluation unit and passed to the Local Control Unit (LCU). Another input to the LCU is the information from the Path Control (i.e. $c^p$), which is further linked to the two higher levels of the architecture. This kind of input is useful for the stimulation of the adaptation in a way that is beneficial from the point of view of the global behaviour of the architecture (see loop $d$ in Figure 5.4). An example of such a control is the management of two types of path, a set of highly adaptive paths, which follow the frequent changes of the data, and a set of more stable paths, which focus on the long term dynamics such as slow data drifts.

An element that is important for the computational paths' adaptation is the Local Memory. Dependent on the implemented adaptation strategy it might be necessary to retrieve some old data samples for the adaptation, as is the case for the moving window technique. In a more advanced scenario, for example the one presented in Chapter 6 or the fully incremental LWPR algorithm presented in Section 3.5.1, it might be sufficient to store some statistics of the data instead of the data samples themselves.

The adaptation techniques themselves are implemented within the Method Control of the Computational Learning Methods (CLM) and of the Pre-Processing Methods (PPM) because they may differ generally from method to method.

The adaptation at this level is activated from the higher level through the Path Control of the Paths module using the $c^p$ signals, which switches the path to the incremental mode (see Figure 5.9). In a particular implementation, this signal may be received by the LCU and further propagated to the PPM and CLM in form of the *mess:adapt!* message (see Figure 5.9).

The choice of adaptation techniques that can be implemented at this position is large. In fact, any concept drift handling technique discussed in Section 3.7 can be used.

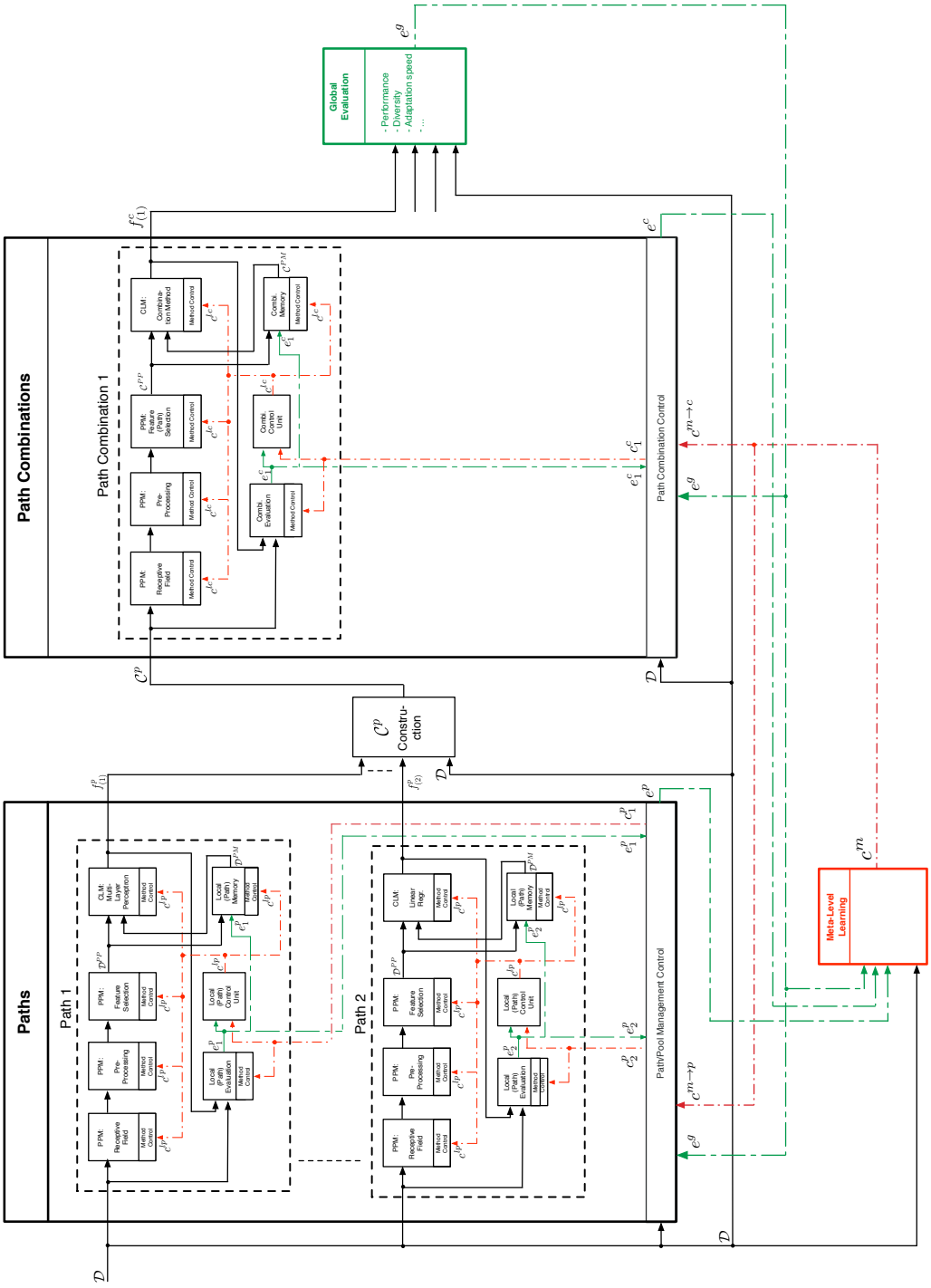



Figure 5.12: Adaptation loops of the architecture and their interactions in detail

### 5.4.2 Path combination level adaptation

The next level of adaptation is the level of Path Combinations (see Figure 5.2 and 5.4). As mentioned in Section 5.3.4, the combinations can be represented in the same way as the computational paths with the only difference being the space in which the combinations are built (the predictions

space $\mathcal{F}^p$ instead of the data space $\mathcal{D}$). A benefit of this representation is that similar adaptation mechanisms as in the case of the path level adaptation, can be applied at the path combination level.

A particularly useful adaptation mechanism that can be implemented at this level is the one presented in Section 4.3.5. This algorithm adapts the combination weights of the ensemble in accordance to the estimated prediction performance of the particular local experts.

### 5.4.3 Meta level adaptation

At the meta level the adaptation has influence on the dynamic behaviour of the whole architecture. As shown in Figure 5.5 the Meta-Level Learning (MLL) module collects the information about the performance of the elements from all levels of the architecture, i.e. from the computational paths $e^p$, path combinations $e^c$ as well as the global performance $e^g$. This information allows one to analyse not only the performance achieved across the different architecture levels but also to estimate the influence of the changes at different positions in the model.

An example of a strategy that uses the meta-level adaptation capabilities and can be implemented within the architecture is AdaBoost [59]. In this case the MLL monitors the performance of the previously deployed computational paths and looks for areas of the data space $\mathcal{D}$ that need improved prediction performance. Once identified, a new receptive field within the Instance Selection Module is deployed and a new computational path is trained for the samples that require better predictions.

## 5.5 Summary

In this chapter a concept for the development of robust and adaptive soft sensors has been presented in the form of an architecture that can be used as a construction plan for the development of soft sensing algorithms.

By involving local learning, ensemble methods, meta-learning and flexible selection of pre-processing and predictive methods, the concept enables one to deal with the issues present in current soft sensor development.

The robustness of the algorithms is achieved by maintaining a diverse set of models consisting of data pre-processing and predictive techniques that are flexibly selected from pools of available methods. The predictions of these models are dynamically combined using ensemble methods. In contrast to the state-of-the-art ensemble methods, there is not only a single ensemble but a set of these maintained. The actual output model delivering the final prediction of the soft sensor is dynamically selected from the set of ensembles.

The architecture also allows reaction to changes in the data as well as to changes of the quality of the data. Another important aspect of the architecture is that through its modular structure it allows the implementation of a wide range of predictive and pre-processing methods, adaptation mechanisms, high-level control techniques, etc.

In order to demonstrate the possibilities of the architecture, an instance of the architecture exploiting many of the provided mechanisms is proposed and evaluated in next chapter.

# Chapter 6

# Complex soft sensing algorithm and soft sensors

## 6.1 Introduction

In this chapter, the theoretical concept for the development of robust and adaptive soft sensors presented in Chapter 5 is instantiated into a complex soft sensing algorithm, which is then used for the development of practical on-line prediction soft sensors. The soft sensing algorithm represents the second step on the thesis pathway shown in Figure 1.3.

The core of the algorithm is the local learning-based technique discussed in Chapter 4. One of the benefits of the algorithm presented in Chapter 4 was its flexibility and the possibility to expand it to a more complex technique in a straight-forward, modular, way. This aspect is fully utilised in this chapter, the key points where the previous method is extended are:

- multiple local experts per receptive field are used
- flexible selection of pre-processing and predictive methods for the local experts is performed
- local expert management using competitive, diversity and cooperative selection is implemented
- complex data management based on local cross-validation and boosting-like approaches is introduced
- adaptation mechanisms at different levels of information processing are used.

The whole algorithm is assembled from available machine learning techniques for data pre-processing, predictive modelling, adaptation and meta-learning and demonstrates that robust and adaptive algorithms can be developed by exploiting and combining the benefits of simple and well-known approaches, such as cross-validation, boosting, or meta-learning, which have been available for many years.

The discussed soft sensing algorithm is applied to develop soft sensors for the same data sets as used in Chapter 4. The soft sensor corresponds to the third step of the thesis pathway as shown in Figure 1.3.

The structure and dependencies between the sections of this chapter are shown in Figure 6.1.

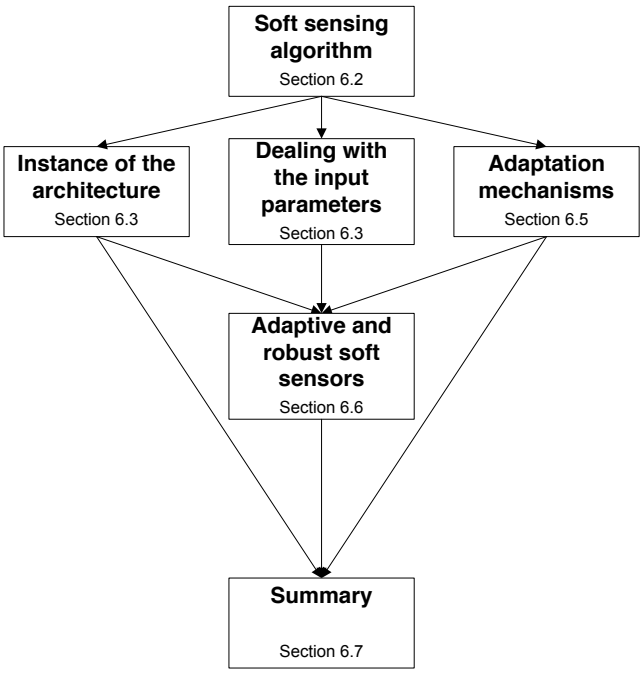



Figure 6.1: The structure of this chapter

## 6.2 Soft sensing algorithm

This section provides an algorithmic description of the proposed complex soft sensing technique.

The algorithm can be split into three different parts as described by the following three functions:

- trainPhaseOne($\mathcal{D}^{hist}$)
- trainPhaseTwo($\mathcal{D}^{hist}$)
- predictAndAdapt($\mathcal{D}^{online}$)

The first two parts are responsible for the training of the soft sensor using the historical data $\mathcal{D}^{hist}$, from which the second one is optional, while the third is used during the on-line phase to obtain the predictions and to adapt the soft sensor given the on-line data $\mathcal{D}^{online}$. The adaptation part is carried out only if the corresponding target value is available. The details of the three methods will be presented in the following sections.

From the perspective of the methodology presented in Figure 4.3, the first two functions represent the second stage of the methodology while the third function is implemented within the third step of the methodology.

Before describing the details of the algorithms, Table 6.1 summarises the input parameters of the soft sensing algorithm and shows the architecture modules that implement the related algorithms.

### 6.2.1 Training phase: Two-step training

The reason for having two training methods is that the algorithm uses a model building approach consisting of two separate steps.

| Module | Parameter | Comments |
|---|---|---|
| Instance Selection Management | $n^{init}$ | The size of the initial window<br>Default: set to $3 * m$ (number of variables of the data set) |
| | $f^{LM}$ | The landmarker computational path<br>Default: PCA (covering $95\%$ of the variance) + MLR |
| Pre-Processing Methods Pool | SF: $n^{smooth}$ | Length of the smoothing filter window<br>Default: $[1, 4, 7, 10]$ |
| | RobPCA: $t^{covVar}$ | Covered variance of the RPCA<br>Default: $[0.90, 0.95, 0.99]$ |
| Computational Learning Methods Pool | MLR | Multi-linear regression<br>Default: no parameters |
| | SVM: $k$ | Size of the Gaussian kernel of the SVM<br>Default: $[0.1, 1, 10, 100, 1000]$ |
| | MLP: $n^{hidden}$ | Number of hidden units of the MLP<br>Default: $[1, 2, 3, 5]$ |
| | LWL: $n^{neighbour}$ | Neighbourhood size of the LWL<br>Default: $[10, 50, 100]$ |
| | LWPR: $d^{init}$ | Initialisation of the distance matrix<br>Default: $[0.1, 1.0, 10.0]$ |
| | LWPR: $w^{gen}$ | Generalisation weights<br>Default: $[0.1, 0.5, 0.8]$ |
| Paths | $n^{LEC}$ | Number of Local Expert Candidates to be trained for each receptive field<br>Default: $100$ |
| | $n^{LE,target}$ | Targeted size of the local expert population in the Path module<br>Default: $10$ |
| | $t^{comp}$ | Threshold for the competitive selection<br>Default: $1.5$ |
| | $t^{div}$ | Threshold for the diversity selection, controlled from MLL<br>Default: initially $0.8$ then adjusted to maintain stable size of the local expert population $n^{LE,target}$ |
| | $\sigma$ | Kernel size of the Gaussian function for the local expert descriptors<br>Default: $10^{-3}$ |
| | $\sigma^{adapt}$ | Kernel size of the Gaussian function for the adaptation of the local expert descriptors<br>Default: $10^{-3}$ |
| Path Combinations | $n^{ens}$ | Number of managed path combinations (ensembles)<br>Default: $10$ |
| Meta-Level Learning | $n^{stepsPerfDescr}$ | Number of iterations for determining the performance descriptors $\mathcal{P}$<br>Default: $5 * n^{par1}$ |
| | $t^{ada3}$ | Threshold (absolute prediction error) for triggering the Type 3 adaptation<br>Default: $0.10$ |
| | $t^{ada4}$ | Threshold (absolute prediction error) for triggering the Type 4 adaptation<br>Default: $0.10$ |
| | $t^{ada5}$ | Threshold (absolute prediction error) for triggering the Type 5 adaptation<br>Default: $0.10$ |

Table 6.1: Summary of the parameters of the soft sensing algorithm

The goal of the first phase is to build an initial knowledge base of the model. After this phase, the model is operational and can be deployed to the on-line phase. The goal of the second, optional, phase is to revise the initial model and adjust it by applying mechanisms for model adaptation,

which will also be applied during the on-line phase. By processing the historical data once more, (i) redundant local experts can be removed; (ii) areas that are not covered well can be covered by new local experts; (iii) local expert descriptors can be adjusted.

A particular benefit of this two-step approach is a smooth transition from the training to the on-line phase. Another benefit is that in this way some important adaptation parameters can be adjusted during the training phase.

### Training phase: First step

The first step is similar to the training approach discussed in Section 4.3. The goals of this step are: (i) splitting the data into several receptive fields; (ii) training a set of local experts for each receptive field; (iii) selecting the local experts who are performing well and are uncorrelated; and (iv) building ensembles from the selected experts.

Algorithm 6.1 gives an overview of the first step of the soft sensor training.

**Algorithm 6.1** trainPhaseOne($\mathcal{D}^{hist}$):

$\mathcal{D}^{RF}, \mathcal{M} \leftarrow$ *buildReceptiveFields(*$\mathcal{D}^{hist}, n^{init}, f^{LM}$*)*;
**for all** $\mathcal{D}_r^{RF} \in \mathcal{D}^{RF}$ **do**
  $P_r \leftarrow$ *buildPerformanceDescriptors(*$\mathcal{D}_r^{RF}, n^{stepsPerfDescr}$*)*;
  $\mathcal{F}_r^{LEC}, \mathbf{e}^{LEC} \leftarrow$ *trainLocalExperts(*$\mathcal{D}_r^{RF}, n^{LEC}, P_r$*)*;
  $\mathcal{F}_r^{LEC} \leftarrow$ *runCompetitiveSelection(*$\mathcal{F}_r^{LEC}, \mathbf{e}^{LEC}, t^{comp}$*)*;
**end for**
$\mathcal{F}^{LEC} \leftarrow \left\{\mathcal{F}_r^{LEC}\right\}_r$;
$\mathcal{D}^{val} \leftarrow \mathcal{D}^{hist}$;
$\mathcal{L} \leftarrow$ *buildLocalExpertDescriptors(*$\mathcal{D}^{val}, \mathcal{F}^{LEC}, \sigma$*)*;
$\mathcal{F}^{LE} \leftarrow$ *runDiversitySelection(*$\mathcal{D}^{val}, \mathcal{F}^{LEC}, \mathcal{L}, t^{div}$*)*;
$\mathcal{F}^{ens}, \mathcal{F}^{ens,winner} \leftarrow$ *buildEnsembles(*$\mathcal{D}^{val}, \mathcal{F}^{LE}, n^{ens}$*)*;

The outcome of this stage is a trained model $\mathcal{F}^{ens,winner}$, which can be deployed as a fully operational soft sensor to the on-line phase.

In the following section the particular functions of the above algorithm will be explained in detail.

**buildReceptiveFields():** The input to this function is:

- $\mathcal{D}^{hist}$: the historical data
- $n^{init}$: the length of the initial window
- $f^{LM}$: the computational path used as landmarker

The approach for receptive fields building is the same as described in Section 4.3.1. An example of splitting the data into receptive fields is demonstrated in Figure 6.2, where the assignment of data points to three receptive fields is shown.

Additionally to the receptive fields, the function also constructs meta descriptors of the receptive fields. The meta descriptors have the following format:

$$M_r = \left[\left\|\mathcal{D}_r^{RF}\right\|, j_r, \text{perf}(f_r^{LM})\right], \tag{6.1}$$

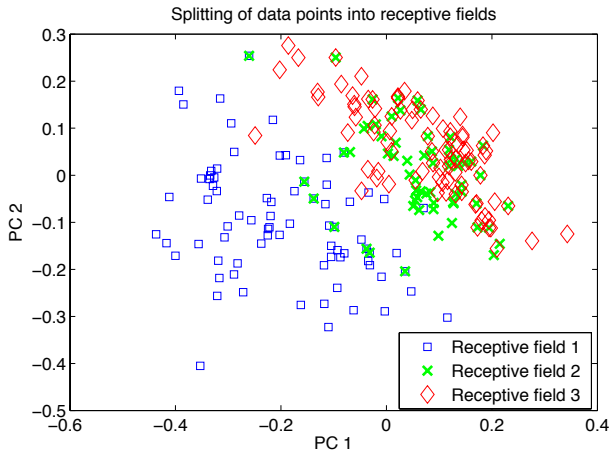



Figure 6.2: Splitting of data points into three receptive fields (displayed after projection to two-dimensional space by means of principal component analysis)

where $\left\|\mathcal{D}_r^{RF}\right\|$ is the number of data samples in the $r$-th receptive field, $j_r$ is the number of variables of the receptive field data after the local pre-processing using, for example, the PCA, and perf$(f^{LM})$ is the quadratic error of the prediction of the landmarker associated with the receptive field. This meta descriptor will allow one to find similar receptive fields during the on-line phase and to inherit their performance descriptors $P$, which will in turn save computational resources during the on-line phase. The meta descriptors of the receptive fields shown in Figure 6.2 are illustrated in Table 6.2:

| | $\left\|\mathcal{D}_r^{RF}\right\|$ | $j_r$ | perf$(f_r^{LM})$ |
|---|---|---|---|
| $M_1$ | 79 | 4 | 0.0168 |
| $M_2$ | 63 | 4 | 0.0052 |
| $M_3$ | 87 | 5 | 0.0017 |

Table 6.2: Example of a meta descriptor $M$

At its output this function provides a set of data object $\mathcal{D}^{RF} = \left\{\mathcal{D}_r^{RF}\right\}_r$ each of which contains data for one receptive field (partition of the historical data) and set of meta descriptors $\mathcal{M} := \{M_r\}_r$ of the receptive fields.

**buildPerformanceDescriptors():** The input to this function is:

- $\mathcal{D}_r^{RF}$: the data for the current receptive field
- $n^{stepsPerfDescr}$: the number of iterations to calculate the performance descriptor
- The pre-processing and computational learning methods and their parameters (from PPMP and CLMP respectively)

The goal of the performance descriptor $P$ is to build a map of the performances of the pre-processing and predictive techniques for the given receptive field. The first step of the descriptor calculation is to draw random samples from the space of all possible parameters of the pre-processing and predictive techniques, which are available in the two method pools (PPMP and CLMP) and building computational paths (see Section 5.3.3 for detailed description of

computational paths) using the drawn methods and their parameters. This action is performed $n^{stepsPerfDescr}$-times. The built computational paths are then trained and their prediction performance is estimated using two-fold cross-validation. Using this method is a trade-off between the generalisation performance estimation accuracy and the computational requirements.

The parameter ranges of the techniques (e.g. the number of hidden units range of an MLP) to be tested have to be defined by the user. However, the pre-defined parameter ranges listed in Table 6.1 are selected in such a way that they can deal with typical tasks. Nevertheless, by changing the ranges or adding/removing some methods for either of the pools using a-priori knowledge (i.e. experience with the application of the methods to similar data sets), can be beneficial to limit the computational requirements and/or improve the performance.

Figure 6.3 shows the performance descriptors for the three receptive fields shown above. An interesting fact are the differences between the descriptors for the different RF. One can, for example, see that for the first RF, the SVMs are dominant in the performance (almost independently of the pre-processing) and the MLR and MLP predictive methods show very bad performance. On the other hand, the second RF shows very strong dependency on the pre-processing and indicates high performance of the LWPR, LWL and MLP models. In the case of this RF it seems to be important to use RobPCA covering $95\%$ of the variance together with a medium length smoothing filter. The third RF shows similar patterns as the first one, with high performance of SVM models and very low performance of the MLPs.

Figure 6.4 shows an example of the output of this function, in this case built for MLR, MLP, RBFN and LWL computational learning methods in combination with PCA and Smoothing Filter (SF) pre-processing. The figure shows that, in this case, the best performing computational paths are those based on LWL and four/seven samples a long SF combined with a PCA (see the peaks in Figure 6.4).

As mentioned above, the output of this function is the performance descriptor $P$ for the current receptive field. The descriptors are stored in the MLL module and will later be used to generate parameters for the local experts (see function *trainLocalExperts()*).

**trainLocalExperts():** The input to this function is:

- $\mathcal{D}_r^{RF}$: the data for the current receptive field
- $\mathcal{D}^{hist}$: the historical data
- $P_r$: the performance descriptor for the current receptive field
- $n^{LEC}$: the number of LEC to be built

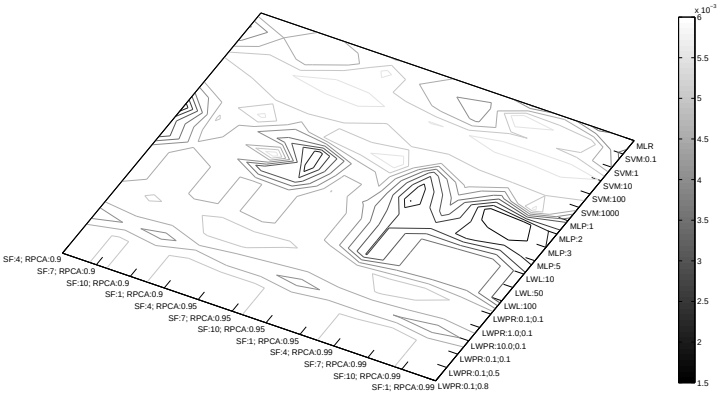
The set of models trained in this function is referred to as *Local Expert Candidates* (LECs):

$$\mathcal{F}^{LEC} := \left\{ f_{(k)}^{LEC} \right\}_{k=1}^{n^{LEC}}, \tag{6.2}$$

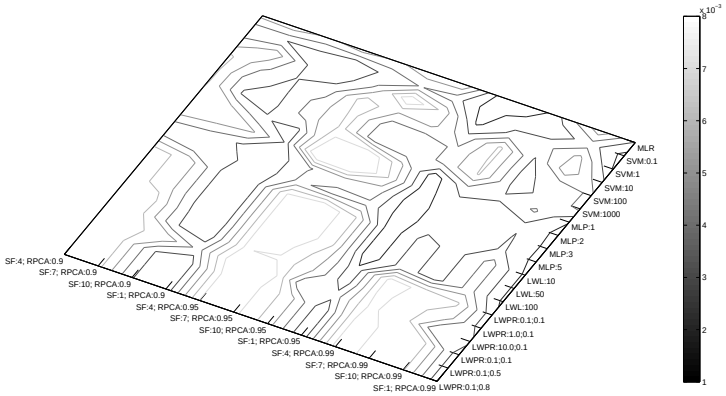
The pre-processing methods, predictive algorithm and the parameters for each of the LEC are obtained by drawing samples from the performance distribution $P_r$.

In order to further boost the diversity and increase the outlier robustness of the trained models, each of the LEC is trained on a random sample (drawn with replacement) of the receptive field data $\mathcal{D}_r^{RF}$. At this point the probability of the data points to be drawn is uniform:

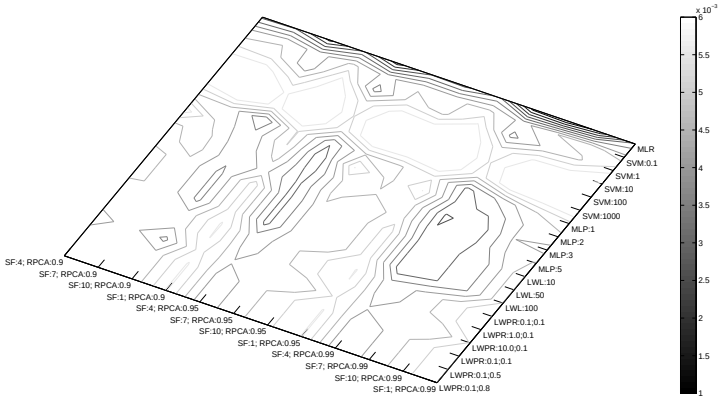
$$\mathbf{p} := \frac{1}{\|\mathcal{D}_r^{RF}\|}\mathbf{u}, \tag{6.3}$$




(a) First receptive field




(b) Second receptive field




(c) Third receptive field

Figure 6.3: Performance descriptors for three receptive fields from a single data set (the intensity of the gray contours indicates the performance level, i.e the lighter the contour, the higher the performance level of the method)

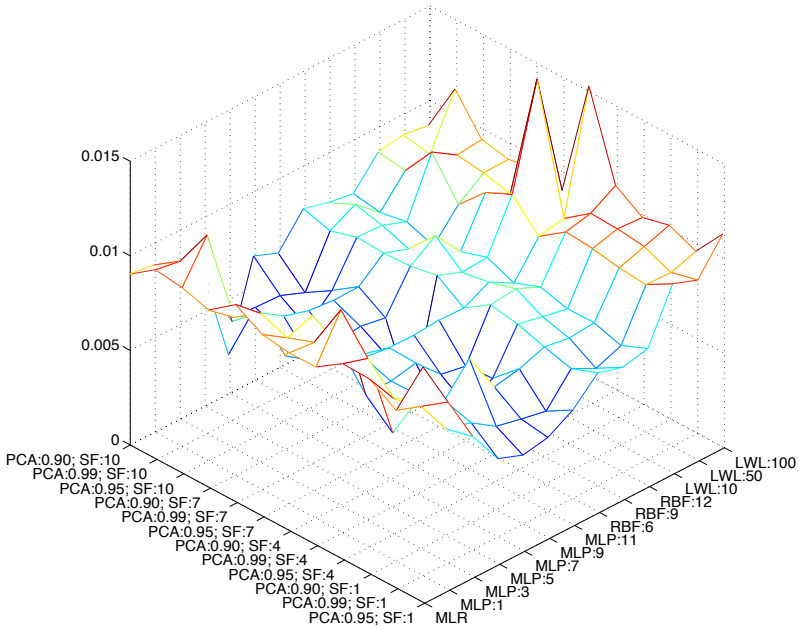



Figure 6.4: Example of a performance descriptor $P_r$

where $\mathbf{u}$ is a unity vector, $\left\|\mathcal{D}_r^{RF}\right\|$ is the number of data points in the current receptive field, which is also the length of the vector $\mathbf{p}$.

Furthermore, in order to effectively exploit the limited available data, each LEC is trained using two-fold cross-validation technique. For further processing, the two sub-models $f^{subLEC}$ resulting from the cross-validation are combined using the robust Least-Means-Square (RLMS) technique. The RLMS-based combination method turned out to be beneficial for the sub-model combination during preliminary experiments with the process industry data sets and proceeds as follows:

$$
\begin{aligned}
f^{LEC} &= \sum_{i=1}^{2} u_i f^{subLEC} \\
\text{with} \quad & \mathbf{u} = r(A, \mathbf{y}^{\mathbf{val}}) \\
\text{with} \quad & A = \begin{bmatrix} f_1^{subLEC}(\mathbf{x}_1^{val}) f_2^{subLEC}(\mathbf{x}_1^{val}) \\ \vdots \\ f_1^{subLEC}(\mathbf{x}_n^{val}) f_2^{subLEC}(\mathbf{x}_n^{val}) \end{bmatrix} \\
\text{with} \quad & \mathcal{D}^{val} = \left\{ \mathbf{x}_i^{val}, y_i^{val} \right\}_{i=1}^{n} = \left( \mathcal{D}^{hist} - \mathcal{D}_i^{RF} \right)
\end{aligned}
\tag{6.4}
$$

where $r()$ is the Matlab *robustfit* function, which is a robust version of the LMS algorithm, $A$ is a matrix of the predictions of the two sub-models for all samples from the validation set $\mathcal{D}^{val}$. The validation set consists of all historical data samples that are not present in the current receptive field.

At the output this function provides a set of LECs trained for the current receptive field $\mathcal{F}_i^{LEC}$ as well as a vector of errors $\mathbf{e}^{LEC} = \left[e_1^{LEC}, \ldots, e_{n^{LEC}}^{LEC}\right]$. The prediction error $\mathbf{e}^{LEC}$ for each LEC is the average error of the errors of the cross-validation sub-models. The error function itself is the Local (Path) Evaluation implemented in each computational path (see Figure 5.6). In the

case of this implementation, the measure is a combination of the MSE and correlation coefficient.

**runCompetitiveSelection():** The input to this function is:

- $\mathcal{D}_i^{hist}$: the data for the current receptive field
- $\mathbf{e}^{LEC}$: the prediction errors of the LECs
- $t^{comp}$: the threshold value for the competitive selection

The first reduction of the number of LECs is performed by means of competitive selection. The aim of the competitive selection is to remove LECs with low performance. The selection is carried out inside each of the receptive fields and therefore at this stage only LECs from the same receptive field are competing with each other. As mentioned above, the performance is measured in terms of the Local (Path) Evaluation function (see Figure 5.6) and is based on the cross-validation performance estimation. In order to pass this selection, the LECs have to fulfil the following condition:

$$\begin{aligned} \forall_{k=1:n^{LEC}}: \quad & e_k^{rel} \leq t^{comp}, \\ \text{with} \quad & e_k^{rel} = \frac{e_k}{e_{min}}, \\ \text{and} \quad & e_{min} = \underset{k=1:n^{LEC}}{\operatorname{argmin}} (e_k), \end{aligned} \tag{6.5}$$

where $e_k^{rel}$ is the relative error of the $k$-th LEC and $e_{min}$ is the LEC with minimal error, i.e. best performing LEC.

Figure 6.5 visualises the selection process for the LECs from the three receptive fields. The figure shows that the number of LECs that pass the selection can vary from receptive field to receptive field significantly. For example, in the first receptive field there are many (24) LECs that pass the selection, see Figure 6.3(a). If not treated, this could cause a bias in the final prediction later on. Preventing such a potential danger is the task of the diversity selection step.

The LECs that pass the selection are the output of this function. Those that do not pass are removed from the model.

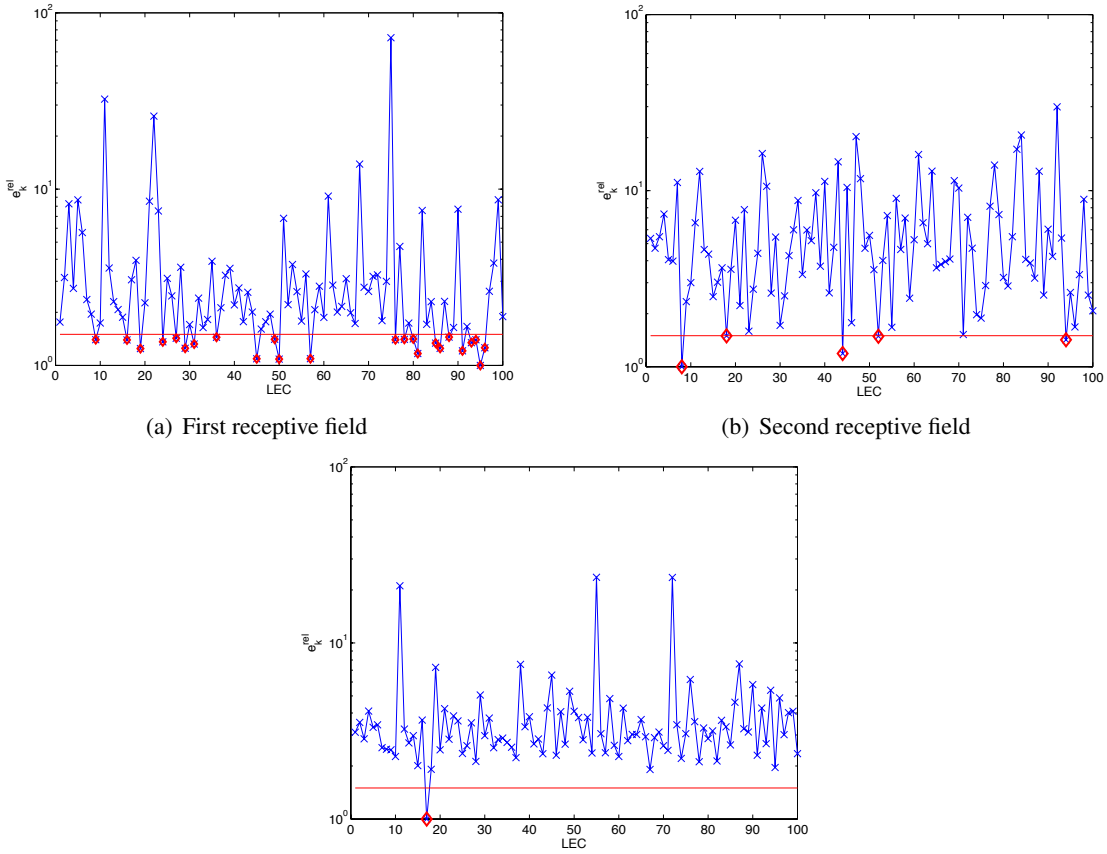
**buildLocalExpertDescriptors():** The input to this function is:

- $\mathcal{D}^{val}$: the validation data
- $\mathcal{F}^{LEC}$: a set of LECs merged over all receptive fields
- $\sigma$: the width of Gaussian kernel function for the Parzen distribution estimation

The algorithm for the calculation of the local expert descriptors is equal to the algorithm presented in Section 4.3.3. The validation data $\mathcal{D}^{val}$ used for the calculation of the descriptors are, in the case of this algorithm, all historical, i.e. training, data. In general, there is potentially a danger of overfitting and too optimistic performance estimation when using the training data for the validation. However, in the local learning case, this effect is limited because the LEC are trained using only partitions of the historical data and will get too optimistic performance indications only for these limited parts of the training data.

The outcome of this function is a set of two-dimensional maps (see Figure 4.5):

$$\mathcal{L}_k = \{L_{k,j}\}_{j=1}^{m}, \tag{6.6}$$




(a) First receptive field

(b) Second receptive field

(c) Third receptive field

Figure 6.5: Performance selection for the three receptive fields used throughout this section

where $m$ is the number of variables after the local, i.e. receptive field related pre-processing of the LECs. These are calculated for each of the LECs. These descriptors model the relative performance of the particular LEC and as such can be used for the estimation of the prediction performance of the LECs, in *performPrediction()*.

**runDiversitySelection():** The input to this function is:

- $\mathcal{D}^{val}$: the validation data
- $\mathcal{F}^{LEC}$: the set of Local Expert Candidates
- $\mathcal{L}$: the local expert descriptors
- $t^{div}$: the threshold value for the diversity selection

At this stage LECs from all receptive fields are joined together and compete with each other. The goal of this selection step is to filter out LECs that are correlated. The correlation level measurement is based on the validation data set $\mathcal{D}^{val}$, and the local expert descriptors $\mathcal{L}$. The descriptors are read at positions defined on one hand by the input data of the validation data $\mathbf{x} \in \mathcal{D}^{val}$ and the predictions $f^{LEC}(\mathbf{x})$ of the LECs on the other hand (for details of the calculation of the weights vector see Equation 4.14 and 4.15 in Section 4.3.4). In this way weights vectors $\omega_m$

for the LECs are obtained. It is among these vectors where the correlation is measured. The reason for using this approach, and not the prediction vectors directly, for the correlation calculation is that in this way the similarity of the predictions as well as the descriptors is measured. In order to obtain correlated weight vectors, the predicted values and the local expert descriptors have to be correlated. Since the descriptors reflect the (local) pre-processing and the local experts expertise by using the weights $\omega_m$ for the diversity selection, all of these aspects can be analysed at the same time.

Having the weight vectors $\omega_m$ for all LECs, their correlation is calculated and those having the strongest correlation patterns are removed from the set of LECs $\mathcal{F}^{LEC}$:

$$maxCorr = \underset{k}{\operatorname{argmax}} \sum_{l,l\neq k} \left( corr(\omega_k, \omega_l) > t^{div} \right) \tag{6.7}$$

$$\mathcal{F}^{LEC} = \left\{ \mathcal{F}^{LEC} - f^{LEC}_{(maxCorr)} \right\}, \tag{6.8}$$

where $corr()$ is the correlation coefficient between the two vectors. This step is iterated as long as the sum is larger than one, or there is only one LEC left (see functions *trainPhaseTwo()* and *predictAndAdapt()*).

Further on, the threshold $t^{div}$ will be manipulated in order to maintain a stable size of the local expert population.

Figure 6.6 shows the correlation matrix of the weight vectors of the LECs that passed the competitive selection. In Figure 6.5(a) it was shown that a larger number of LECs passed the competitive selection in the first RF, therefor one can expect high correlation among these LECs. This is confirmed in Figure 6.6, where one can see high correlation among the first 24 LECs. After the diversity selection, there are only the following eight LECs left: $2, 13, 25, 26, 27, 28, 29, 30$. As one can see in the previous list, there are only two $(2, 13)$ from the original 24 LECs from the first RF left. The correlation matrix of these LECs is shown in Figure 6.7.

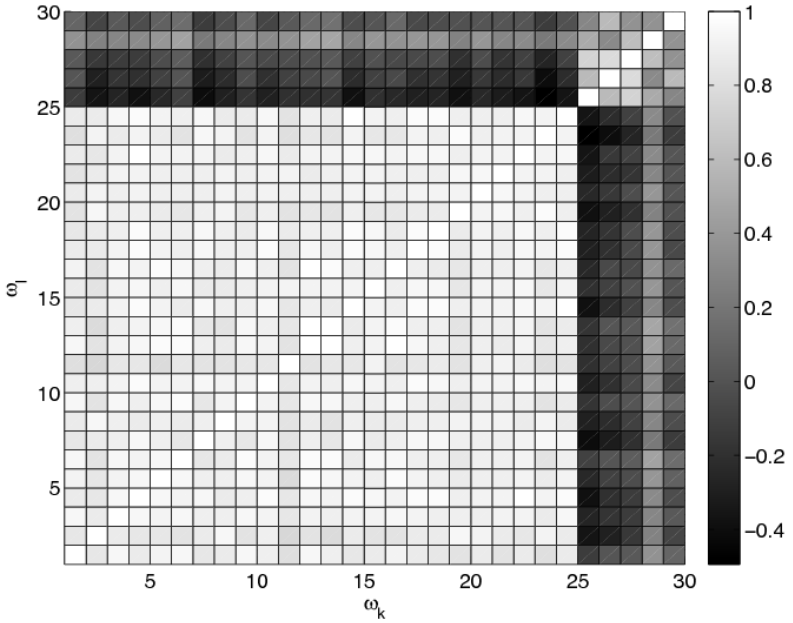



Figure 6.6: Correlation coefficients of all LECs (from the three different receptive fields)

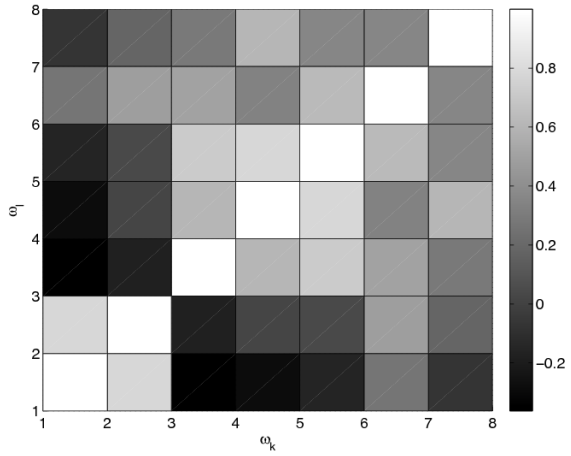



Figure 6.7: Correlation coefficients of the LECs remaining after the diversity selection

The LECs that pass this selection step are called Local Experts (LEs) and are output by this function as a set of local experts $\mathcal{F}^{LE}$.

**buildEnsembles():** The input to this function is:

- $\mathcal{D}^{val}$: the validation data
- $\mathcal{F}^{LE}$: the set of Local Experts
- $n^{ens}$: the number of ensembles to be built

The task of this function is to perform the third selection step, namely the co-operative selection. The goal of this selection is to build teams of LEs that co-operatively achieve high performance. In contrast to the previous two selection steps, the LEs that are not selected for any ensemble are kept in the model since they can be useful at a later stage. The ensemble building proceeds as follows:

Based on the evaluation data set $\mathcal{D}^{val}$ used earlier, the best performing LE ($f^{LE}_{(min)}$) in terms of the Local (Path) Evaluation error measure (here, presented in terms of the mean squared error) is selected:

$$\begin{aligned} f^{LE}_{(min)} &= \underset{m=1:n^{LE}}{\operatorname{argmin}} \left( \sum_{\{\mathbf{x}_i, y_i\} \in \mathcal{D}^{val}} (f^{LE}_{(m)}(\mathbf{x}_i) - y_i)^2 \right) \\ \text{with} \quad \mathcal{D}^{val} &= \{\mathbf{x}_i, y_i\}_{i=1}^{n^{val}}, \end{aligned} \tag{6.9}$$

where $n^{val}$ is the number of data points in the validation data set and $n^{LE}$ the number of local experts. After finding the best performing LE, it is put to the first ensemble $\mathcal{F}^{ens}_1$:

$$\mathcal{F}^{ens}_1 = \left\{ f^{LE}_{(min)} \right\}. \tag{6.10}$$

Next, the Local Experts that most improve the ensembles prediction are iteratively added to the ensemble:

$$f_{(min)}^{LE} = \underset{\substack{m=1:n^{LE} \\ f_{(m)}^{LE} \notin \mathcal{F}_1^{ens}}}{\operatorname{argmin}} \left( \sum_i \left( c\left(\mathcal{F}_{temp}^{ens}(\mathbf{x}_i)\right) - y_i \right)^2 \right) \tag{6.11}$$

$$\text{with } \mathcal{F}_{temp}^{ens} = \left\{ \mathcal{F}_1^{ens}, f_{(m)}^{LE} \right\}$$

$$\mathcal{F}_1^{ens} = \left\{ \mathcal{F}_1^{ens}, f_{(min)}^{LE} \right\}, \tag{6.12}$$

where $c(\mathcal{F}_{temp}^{ens})$ is a combination function that forms the combined prediction of the ensemble (for detail on the combination function see Section 4.3.4). The procedure described in Equations 6.11 is repeated as long as adding a new member to the ensemble improves the performance.

The above procedure is repeated $n^{ens}$-times to build the required number of ensembles. In this particular implementation, the different combinations are varied by changing the initial LE, which is selected randomly for the ensembles $\mathcal{F}_2^{ens}, \ldots, \mathcal{F}_{n^{ens}}^{ens}$.

The set of ensembles built using the approach discussed above is further extended by some special ensembles, these include the full combination, i.e. ensemble including all local experts:

$$\mathcal{F}_{n^{ens}+1}^{ens} = \left\{ f_{(k)}^{LE} \right\}_{k=1}^{n^{LE}}. \tag{6.13}$$

Other special ensembles are the single local experts:

$$\forall_{k=1:n^{LE}} : \mathcal{F}_{n^{ens}+1+k}^{ens} = \left\{ f_{(k)}^{LE} \right\}. \tag{6.14}$$

The output of this function is on one hand a set of ensembles $\mathcal{F}^{ens}$, and on the other the best performing ensemble $\mathcal{F}^{ens,winner}$, which is the model used for making the predictions in the function *makePrediction()*.

### Training phase: Second step

The performance of the resulting soft sensor can be further improved by applying the second step of the training algorithm. This step is also applied to the historical training data and is similar to the on-line part of the algorithm (compare Algorithm 6.2 with Algorithm 6.4):

The first step of the algorithm is getting the predictions of the soft sensor using *makePrediction()*. After this, the target value $y_i$ can be used for the calculation of the performance feedback and adaptation. As discussed in Chapter 5, one of the benefits of the architecture is the provision of several adaptation mechanisms. During this training step three different methods are used for the adaptation of the soft sensor:

- Type 1 adaptation: low-level adaptation of the computational paths $\mathcal{F}^{LE}$ (loop a in Figure 5.4)
- Type 2 adaptation: adaptation of the local expert descriptors $\mathcal{L}$ (loop b in Figure 5.4)
- Type 5 adaptation: complex adaptation involving launching of new receptive fields (loop d in Figure 5.4).

As shown above, the low level adaptation techniques, i.e. the adaptation of the computational paths (Type 1) and the adaptation of the local expert descriptors (Type 2), are performed on a

**Algorithm 6.2** trainPhaseTwo($\mathcal{D}^{hist}$):

**for all** $\mathbf{x}_i \in \mathcal{D}^{hist}$ **do**
  $y^p \leftarrow$ *makePrediction*($\mathbf{x_i}, \mathcal{F}^{ens,winner}, \mathcal{L}$);
  $\mathcal{F}^{LE} \leftarrow$ *runType1Adaptation*($\{\mathbf{x}_i, y_i\}, \mathcal{F}^{LE}$);
  $\mathcal{L} \leftarrow$ *runType2Adaptation*($\{\mathbf{x}_i, y_i\}, \mathcal{F}^{LE}, \mathcal{L}, \sigma^{adapt}$);
  $\mathcal{D}^{buffer} \leftarrow \left\{\mathcal{D}^{buffer}, \{\mathbf{x}_i, y_i\}\right\}$;
  **if** $|y^p - y_i| > t^{ada5}$ **then**
    $\mathcal{F}^{LE}, \mathcal{F}^{ens}, \mathcal{F}^{ens,winner}, \mathcal{P}, \mathcal{M}, \mathcal{L}, t^{div} \leftarrow$
    *runType5Adaptation*($\mathcal{D}^{buffer}, \mathcal{F}^{LE}, \mathcal{P}, \mathcal{M}, n^{init}, f^{lm}, n^{stepsPerfDescr}, \ldots$
    $\ldots, n^{LEC}, t^{comp}, t^{div}, \sigma, n^{ens}$);
  **end if**
**end for**

sample-by-sample basis. These techniques do not require storing any of the past data samples. In contrast to these, the Type 5 adaptation mechanism, which involves launching of new receptive fields, is performed only on-demand when the performance of the soft sensor drops below the defined threshold $t^{ada5}$. For this type of adaptation a set of data, here indicated as $\mathcal{D}^{buffer}$, has to be provided. In general this can be implemented as a First In First Out (FIFO) buffer with limited size, where the size depends on the size of the receptive fields, but in the case of the training algorithm, which uses the historical data available in batch mode, a subset of the data can simply be used.

In the following paragraphs, the functions from Algorithm 6.2 will be described in more detail.

**makePredictions():** The input to this function is:

- $\mathbf{x}_i$: the input data sample for which to make the prediction
- $\mathcal{F}^{ens,winner}$: the current model providing the output prediction $y^p$
- $\mathcal{L}$: the local expert descriptors

The procedure for the calculation of the predictions is a weighted sum of the predictions of the $\mathcal{F}^{ens,winner}$ ensemble members, where the weights are read from the local expert descriptors $\mathcal{L}$. The details of the technique are described in Section 4.3.4.

On its output this function provides the prediction $y^p$ for the current data point $\mathbf{x}_i$.

**runType1Adaptation():** The input to this function is:

- $\{\mathbf{x}_i, y_i\}$: the current input-output data point
- $\mathcal{F}^{LE}$: the set of Local Experts to be adapted

This method implements the adaptation capability at the lowest level of the architecture that involves the adaptation of the computational path (see Figure 5.2), which corresponds to *loop a* in Figure 5.4. In the soft sensing algorithm discussed in this chapter, this functionality is available only with computational paths using the LWPR computational learning method. As demonstrated in the experiments in Section 4.4, this technique provides a powerful and efficient incremental learning technique, which is exploited in this adaptation mechanism.

After the adaptation, the function outputs the adapted local experts $\mathcal{F}^{LE}$.

**runType2Adaptation():** The input to this function is:

- $\{\mathbf{x}_i, y_i\}$: the current input-output data point
- $\mathcal{F}^{LE}$: the set of Local Experts to be adapted
- $\mathcal{L}$: the local expert descriptors
- $\sigma^{adapt}$: the size of the adapted neighbourhood in the local expert descriptors

This adaptation method corresponds to the adaptation *loop b* in Figure 5.4. The adaptation mechanism modifies the local expert descriptors $\mathcal{L}$ based on the relative performance of the local experts and was applied in Chapter 4 as the adaptation method for the LASSA algorithm. The changes of the descriptors have an influence on the combination weights $v$ in the path combinations (see Equation 4.15 for details). This adaptation method can also be performed sample-by-sample and does not require the storage of any past data. The details of the methods are discussed in Section 4.3.5.

The output of this method are the adapted local expert descriptors $\mathcal{L}$.

**runType5Adaptation():** The input to this function is:

- $\mathcal{D}^{buffer}$: the batch of collected input-output data
- $\mathcal{F}^{LE}$: the set of Local Experts
- $\mathcal{P}$: the performance descriptor for the receptive fields
- $\mathcal{M}$: the receptive fields' meta descriptors
- $n^{init}$: the length of the initial window for *buildReceptiveFields()*
- $f^{lm}$: the computational path used as landmarker for *buildReceptiveFields()*
- $n^{stepsPerfDescr}$: the number of iterations for *buildPerformanceDescriptorsOnline()*
- $n^{LEC}$: the number of LECs to be built for *trainLocalExpertsOnline()*
- $t^{comp}$: the threshold value for the competitive selection for *runCompetitiveSelection()*
- $\sigma$: the width of Gaussian kernel function for the Parzen distribution estimation for *buildLocalExpertDescriptors()*
- $t^{div}$: the threshold value for the diversity selection for *runDiversitySelection()*
- $n^{ens}$: the number of ensembles to be built for *buildEnsembles()*

The number of inputs and outputs of this function indicates its complexity. The goal of this method is to launch a new receptive field, train and select new local experts and build new ensembles. Many of these steps can be done by resorting to the functions discussed above.

The first step of the algorithm is the estimation of receptive fields in the $\mathcal{D}^{buffer}$ data. This data consist of a set of collected on-line data points, where the number of the points is dependent on the size of the buffer[2]. After running the *buildReceptiveFields()* function, the most interesting

[2]In this implementation, the buffer size is fixed to 150 data samples (which, for a 30 dimensional data set using double precision, corresponds to ca. 35kByte of memory).

**Algorithm 6.3** runType5Adaptation():

$\mathcal{D}^{temp}, \mathcal{M}^{temp} \leftarrow$ *buildReceptiveFields(*$\mathcal{D}^{buffer}, n^{init}, f^{lm}$*)*;
$\mathcal{D}^{RF,online} \leftarrow \mathcal{D}^{temp}_{end}$;
$\mathcal{D}^{RF} \leftarrow \{\mathcal{D}^{RF}, \mathcal{D}^{RF,online}\}$;
$M^{online} \leftarrow \mathcal{M}^{temp}_{end}$;
$\mathcal{M} \leftarrow \{\mathcal{M}, M^{online}\}$;
$P^{online} \leftarrow$ *buildPerformanceDescriptorsOnline(*$\mathcal{D}^{RF,online}, \mathcal{M}, M^{online}, \mathcal{P}, n^{stepsPerfDescr}$*)*;
$\mathcal{P} \leftarrow \{\mathcal{P}, P^{online}\}$;
$\mathcal{F}^{LEC,online}, \mathbf{e}^{LEC,online} \leftarrow$ *trainLocalExpertsOnline(*$\mathcal{D}^{RF,online}, P^{online}, \mathbf{e}^{p}, n^{LEC}$*)*;
$\mathcal{F}^{LEC,online} \leftarrow$ *runCompetitiveSelection(*$\mathcal{F}^{LEC,online}, \mathbf{e}^{LE,online}, t^{comp}$*)*;
$\mathcal{L}^{online} \leftarrow$ *buildLocalExpertDescriptors(*$\mathcal{D}^{val}, \mathcal{F}^{LEC,online}, \sigma$*)*;
$\mathcal{L} \leftarrow \{\mathcal{L}, \mathcal{L}^{online}\}$;
$\mathcal{F}^{LEC} \leftarrow \{\mathcal{F}^{LE}, \mathcal{F}^{LEC,online}\}$;
$\mathcal{F}^{LE} \leftarrow$ *runDiversitySelection(*$\mathcal{D}^{val}, \mathcal{F}^{LEC}, t^{div}, \mathcal{L}$*)*;
**if** $n^{LE} > n^{LE,target} * 1.1$ **then**
  $t^{div} \leftarrow t^{div} - 0.1$
**end if**
**if** $n^{LE} < n^{LE,target} * 0.9$ **then**
  $t^{div} \leftarrow t^{div} + 0.1$
**end if**
$\mathcal{F}^{ens}, \mathcal{F}^{ens,winner} \leftarrow$ *buildEnsembles(*$\mathcal{D}^{val}, \mathcal{F}^{LE}, n^{ens}$*)*;

is the last receptive field $\mathcal{D}^{temp}_{end}$ because it consists of data points immediately before the error of the soft sensor exceeded the threshold. Next, the performance descriptor $P^{online}$ for this receptive field is calculated using the function *buildPerformanceDescriptorsOnline()*. This function applies a simple meta-learning principle to inherit the performance descriptors from past receptive fields, which makes it slightly different from the *buildPerformanceDescriptors()* function. Having the performance descriptor $P$ for the receptive field, the *trainLocalExpertsOnline()* is used for the training of new LECs. This function also differs minimally from the previous function for LEC training. The only difference is that this function uses the error of the full model as sampling probability for drawing the training samples for the particular LECs. In this way, data points with a high prediction error are more likely to be represented in the training sets of the new LECs. The next steps are the competitive selection among the new LECs using *runCompetitiveSelection()* and the building of local expert descriptors for the LECs that pass the selection using *buildLocalExpertDescriptors()*. After this step, the new LECs $\mathcal{F}^{LEC,online}$ are merged with the rest of the population of the present local experts $\mathcal{F}^{LE}$ and diversity selection among the merged set is performed. This is an effective pruning mechanism since the diversity is checked among all local experts, old and new, and those that are too correlated are removed. This in turn allows one to maintain a stable number of local experts despite the fact that new receptive fields are launched. The next part of the algorithm adjusts the diversity threshold $t^{div}$ in order to keep the number of local experts $n^{LE}$ close to the target population size $n^{LE,target}$. Finally, new ensembles are built using the *buildEnsembles()* method. The following paragraphs describe the new functions.

The effect of this adaptation type on the population of local experts in the Path module can be observed in Figure 6.8. The figure shows that the algorithm manages to maintain a stable population of around ten local experts. Each of the (seven) step changes in the figure corresponds to the execution of the type 5 adaptation mechanism.

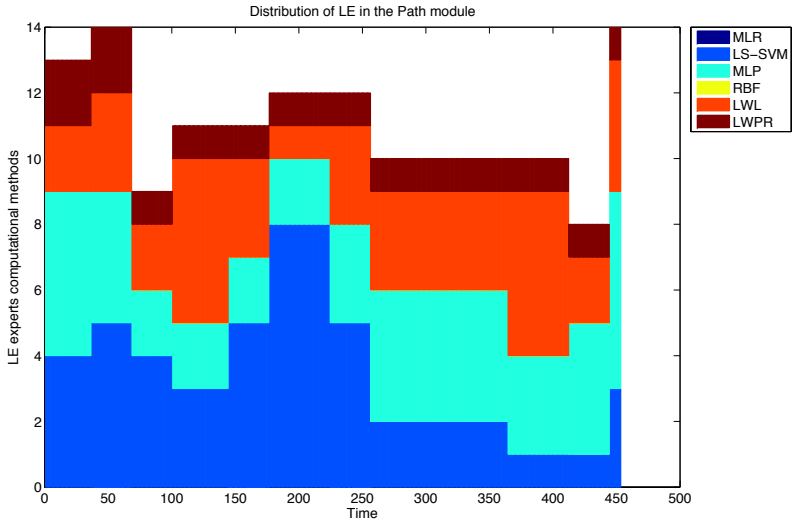



Figure 6.8: An example of the effect of the type 5 adaptation on the population of local experts

**buildPerformanceDescriptorsOnline():** The input to this function is:

- $\mathcal{D}^{RF,online}$: the data for the current receptive field
- $\mathcal{M}$: the meta descriptors of the past receptive fields
- $M^{online}$: the meta descriptors of the current receptive fields
- $\mathcal{P}$: the performance descriptors for the past receptive fields
- $n^{stepsPerfDescr}$: the number of iterations to calculate the performance descriptor
- The pre-processing and computational learning methods and their parameters (from PPMP and CLMP respectively)

In this function the same procedure as in *buildPerformanceDescriptors()* for the calculation of the performance descriptor for the current receptive field is done. However, the difference is that in this function there are only $\frac{n^{stepsPerfDescr}}{10}$ computational paths processed because the processing of the full number of computational paths can be time consuming and thus infeasible for the second step of the training and the on-line phase. This delivers an initial performance descriptor $P^{init}$. The other part of the performance descriptor is inherited from the already available performance descriptors $\mathcal{P}$, which is achieved by searching for the most similar receptive field in the space of of the meta descriptors $\mathcal{M}$. The search is done using the nearest-neighbour technique:

$$M_i = \underset{r=1:\|\mathcal{M}\|}{\operatorname{argmin}} (M_r - M^{online}), \tag{6.15}$$

where $M_i$ is the meta descriptor of the most similar receptive field, the index $i$ is used to identify the corresponding performance descriptor $P_i$, $\|\mathcal{M}\|$ is the number of past receptive fields and the $-$ operator is the Euclidian distance in the space of the meta descriptors.

The final performance descriptor, and the output of this function, for the current receptive field is the average of the two particular descriptors:

$$P^{online} = \frac{1}{2}\left(P^{init} + P_i\right). \tag{6.16}$$

**trainLocalExpertsOnline():** The input to this function is:

- $\mathcal{D}^{RF,online}$: the data for the current receptive field
- $P^{online}$: the performance descriptor for the current receptive field
- $\mathbf{e}^p$: the prediction error of the full model for the data points in $\mathcal{D}^{RF,online}$
- $n^{LEC}$: the number of LEC to be built

This function proceeds the same way as *trainLocalExperts()* with the only difference being the way in which the samples for the training of the LECs are established. In the previous function it was done by uniformly sampling the data points in the receptive field (see Equation 6.3). Motivated by the boosting ensemble method (see Section 3.4), the uniform probability is modified into a probability distribution $\mathbf{p}^{boost}$, which reflects the prediction error $\mathbf{e}^p$, which is the error of the full model on the data points in $\mathcal{D}^{RF,online}$:

$$\begin{aligned} \forall_i : e_i^p &= (\mathcal{F}^{ens,winner}(\mathbf{x}_i^{RF,online}) - y_i^{RF,online})^2 \\ &\text{with } \left\{\mathbf{x}_i^{RF,online}, y_i^{RF,online}\right\} \in \mathcal{D}^{RF,online} \end{aligned} \tag{6.17}$$

$$\mathbf{p}^{boost} = \frac{\mathbf{e}^p}{\sum \mathbf{e}^p}. \tag{6.18}$$

In this way, data points for which the prediction performance of the soft sensor was low will have higher probability to be represented more frequently in the training data for the new LEC.

### 6.2.2 On-line phase: Prediction and adaptation

This phase is similar to the second step of the training phase. A particular difference that can occur in realistic scenarios is that there are fewer target values available and the adaptation is less frequent (see Section 2.3.1 for the discussion of this topic). Nevertheless, the algorithm has a very similar form.

The three more complex adaptation mechanisms in Algorithm 6.4 are arranged in such a way that if the error grows over the threshold value, the first attempt to improve the performance is the simplest adaptation mechanism, namely Type 3 adaptation. If this does not help to improve the performance then next the more complex Type 4 adaptation is tried and only in the case that this is also unsuccessful is the most complex, Type 5, adaptation performed.

The section proceeds with the description of the adaptation functions that were not covered until now.

**runType3Adaptation():** The input to this function is:

- $\mathcal{D}^{buffer}$: the collected on-line data samples
- $\mathcal{F}^{ens}$: the current ensembles from which the best performing one is selected

**Algorithm 6.4** onlinePhase($\mathcal{D}^{online}$):

$y^p \leftarrow$ *makePrediction(*$\mathbf{x}^{online}, \mathcal{F}^{ens,winner}, \mathcal{L}$*)*;
**if** $y^{online}$ available **then**
  *adaptPerformed* $\leftarrow 0$;
  $\mathcal{F}^{LE} \leftarrow$ *runType1Adaptation(*$\mathcal{F}^{LE}, \mathcal{D}^{online}$*)*;
  $\mathcal{L} \leftarrow$ *runType2Adaptation(*$\mathcal{F}^{LE}, \mathcal{D}^{online}, \mathcal{L}$*)*;
  $\mathcal{D}^{buffer} \leftarrow \left\{\mathcal{D}^{buffer}, \mathcal{D}^{online}\right\}$;
  **if** $\left|y^p - y^{online}\right| > t^{ada3}$ **then**
    $\mathcal{F}^{ens,winner} \leftarrow$ *runType3Adaptation(*$\mathcal{D}^{buffer}, \mathcal{F}^{LE}$*)*;
    *adaptPerformed* $\leftarrow 1$;
  **end if**
  **if** $\left|y^p - y^{online}\right| > t^{ada4}$ AND *adaptPerformed* $== 0$ **then**
    $\mathcal{F}^{ens}, \mathcal{F}^{ens,winner} \leftarrow$ *runType4Adaptation(*$\mathcal{D}^{buffer}, \mathcal{F}^{LE}, n^{ens}$*)*;
    *adaptPerformed* $\leftarrow 1$;
  **end if**
  **if** $\left|y^p - y^{online}\right| > t^{ada5}$ AND *adaptPerformed* $== 0$ **then**
    $\mathcal{F}^{LE}, \mathcal{F}^{ens}, \mathcal{F}^{ens,winner}, \mathcal{P}, \mathcal{M}, \mathcal{L}, t^{div} \leftarrow$
    *runType5Adaptation(*$\mathcal{D}^{buffer}, \mathcal{F}^{LE}, \mathcal{P}, \mathcal{M}, n^{init}, f^{lm}, n^{stepsPerfDescr}, n^{LEC}, \ldots$
    $\ldots, t^{comp}, t^{div}, \sigma, n^{ens}$*)*;
  **end if**
**end if**

This adaptation method is part of *loop c* in Figure 5.4. The goal of this adaptation technique is to check the performance of the available path combinations, i.e. ensembles, and to select the best performing one.

$$\mathcal{F}^{ens,winner} := \underset{j}{\operatorname{argmin}} \left( \sum_{i \in \mathcal{D}^{buffer}} \left( \mathcal{F}_j^{ens}(\mathbf{x}_i) - y_i \right)^2 \right) \tag{6.19}$$

The output of this adaptation function is the, currently, best performing path combination $\mathcal{F}^{ens,winner}$.

**runType4Adaptation():** The input to this function is:

- $\mathcal{D}^{buffer}$: the collected on-line data samples
- $\mathcal{F}^{LE}$: the set of Local Experts
- $n^{ens}$: the number of ensembles to be built

Since the triggering of this adaptation mechanism is similar to the Type 3 adaptation and involves the meta level functionality, it can be also seen as *loop c* adaptation mechanism (see Figure 5.4). The function rebuilds the path combinations. This is triggered if the prediction error grows over the $t^{ada4}$ threshold and can be seen as an attempt to improve the performance of the soft sensor relatively *cheaply*. The adaptation is performed by executing the *buildEnsembles()* function discussed above:

$$\mathcal{F}^{ens}, \mathcal{F}^{ens,winner} \leftarrow \textit{buildEnsembles}(\mathcal{D}^{buffer}, \mathcal{F}^{LE}, n^{ens}) \tag{6.20}$$

The outcome of this function is a new set of ensembles $\mathcal{F}^{ens}$ as well as the current output model $\mathcal{F}^{ens,winner}$.

## 6.3 Soft sensing algorithm as an instance of the architecture

This section presents the complex soft sensing algorithm as an instance of the architecture proposed in Chapter 5. Figure 6.9 shows an overview of the architecture instance. The figure shows in which modules the particular functions, descriptors and data objects discussed in Section 6.2 are implemented.

The figure also lists the methods implemented in the pools of pre-processing (PPMP) and computational learning methods (CLMP). For the sake of clarity, the pooling connections between the PPMP, CLMP and the paths are skipped in the figure.

The role of the Expert Knowledge module is to provide the parameters setting of the algorithm (see Table 6.1) to the other modules of the architecture.

## 6.4 Dealing with the input parameters

The number of the input parameters of the proposed soft sensing algorithm, as presented in Table 6.1, may on first sight be overwhelming and interpreted as an obstacle for the application of the algorithm in practical scenarios.

In order to analyse these concerns, the parameters are split into the following classes:

- D: these are parameters that do not need any optimisation and *default* values as stated in Table 6.1 can be used. Due to the flexibility of the proposed algorithm, changing these parameters will not have a strong effect on the performance of the resulting model. These also include the parameters whose default values are defined as ranges in Table 6.1. For these parameters the algorithm provides optimisation mechanisms that automatically select the optimal parameter values.

- R: these parameters can be set according to the available computational *resources*. In general, increasing the value of these parameters leads to better performance at the cost of higher demand for computational resources

- O: these parameters are *optimised* during the run-time of the model and their setting is merely the initial value

- P: *performance* critical parameters, the optimal values of these parameters is task- and data-dependent and should be considered for optimisation for each task.

Using the above classification of the parameters Table 6.3 shows to which class each of the parameters listed in Table 6.1 belongs.

The table shows that many of the parameters can be kept at their default values and do not need to be optimised. This includes the ranges for the pre-processing and computational learning methods in PPMP and CLMP respectively as well as the parameters for the data partitioning. To demonstrate this claim, all of the parameters from class D will be kept constant at their default values in the subsequent experiments.

Two parameters, which should be set-up according to the available computational resources, are $n^{stepsPerfDescr}$ and $n^{LEC}$. The first parameter defines how many iterations are performed

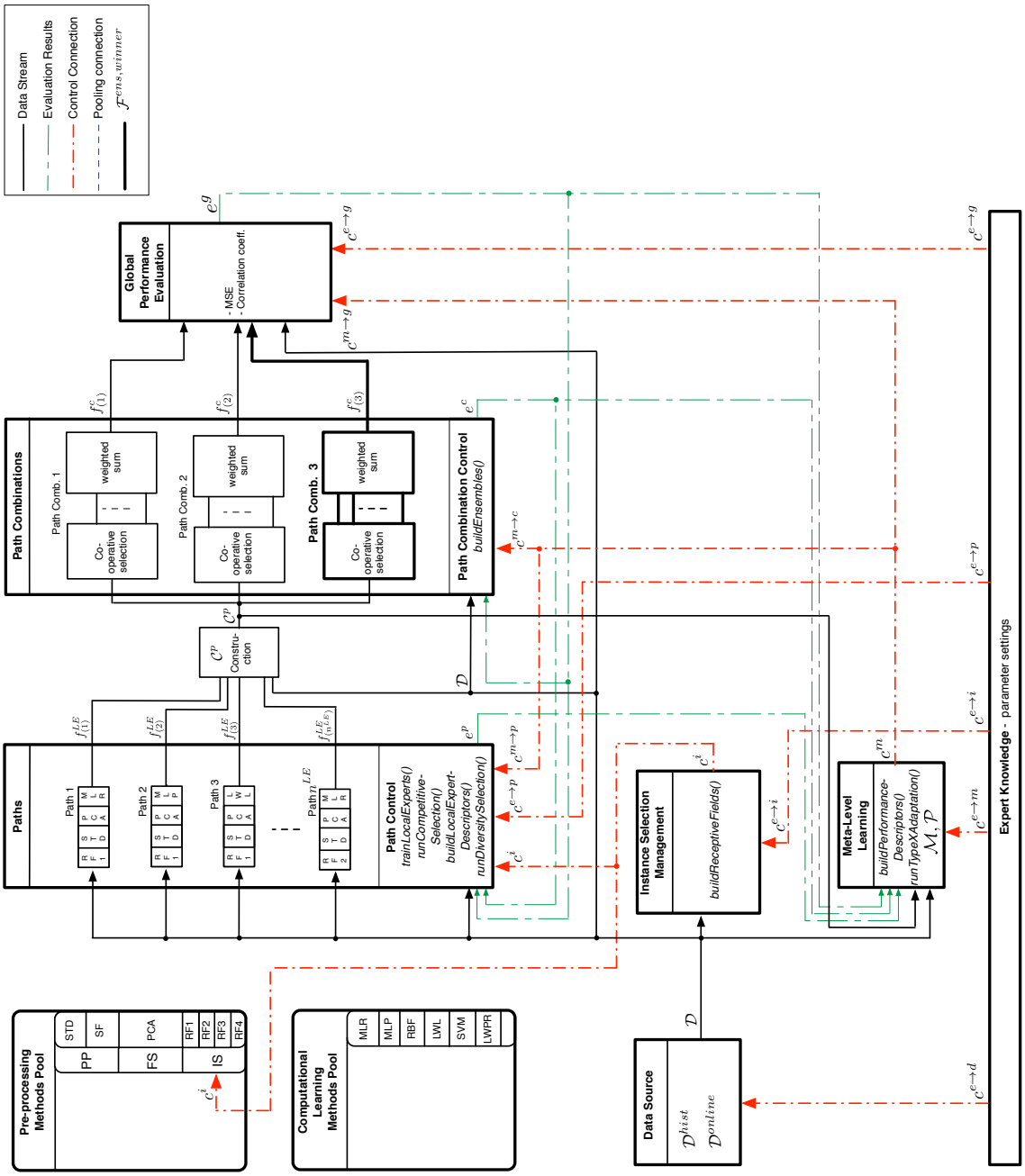



Figure 6.9: The complex soft sensing algorithm as an instance of the architecture from Chapter 5

for the calculation of the performance descriptors for the receptive fields (see Section 6.2.1 for details of the calculation). Higher values of this parameter can be beneficial, especially if the pool of computational methods includes non-deterministic methods, such as ANN with initialisation-dependent performance, and lead to more accurate performance descriptors. This in turn is expected to result in better selection of the pre-processing and computational learning methods for the local experts. The latter parameter defines the number of local expert candidates trained for each receptive field. Training more LECs increases the chances that well performing LECs are trained and pass the competitive selection (see Section 6.2.1 for details).

| Group | Parameter |
|---|---|
| D | $n^{init}$ |
| | $f^{LM}$ |
| | SF: $n^{smooth}$ |
| | RobPCA: $t^{covVar}$ |
| | SVM: $k$ |
| | MLP: $n^{hidden}$ |
| | RBF: $n^{hidden}$ |
| | LWL: $n^{neighbour}$ |
| | LWPR: $d^{init}$ |
| | LWPR: $w^{gen}$ |
| | $t^{comp}$ |
| | $n^{ens}$ |
| R | $n^{LEC}$ |
| | $n^{stepsPerfDescr}$ |
| O | $t^{div}$ |
| P | $n^{LE,target}$ |
| | $\sigma$ |
| | $\sigma^{adapt}$ |
| | $t^{ada3}$ |
| | $t^{ada4}$ |
| | $t^{ada5}$ |

Table 6.3: Input parameters of the complex soft sensing algorithm and their allocation to the different parameter classes

There is also one parameter that is optimised on-line during the run-time of the algorithm. The $t^{div}$ parameter is controlled in order to maintain a stable size of the local experts in the Paths module. The setting of this parameter is only an initial value, which can influence the time needed for the stabilisation of the population and as such in general does not need to be changed from its default value because its value is set-up during the second step of the soft sensor training.

The most critical parameters from the viewpoint of the soft sensor developer are the class P parameters as their values are task and/or data dependent. In the case of the proposed complex soft sensing algorithm, these parameters are:

- The target population size $n^{LE,target}$: the influence of this parameter will be further investigated in the experiments in Section 6.6.1.

- The kernel size for the local expert descriptors and their adaptation $\sigma$ and $\sigma^{adapt}$ respectively: the impact of these parameters on the performance for the three data sets was analysed in Section 4.4 and because the same algorithm is the core of the complex soft sensing algorithm a similar influence on the performance can be expected. For the experiments in this chapter, the default value without any further optimisation is used for both of the parameters. The influence of the parameter value is visualised in Figure 4.5.

- The adaptation thresholds, $t^{ada3}$, $t^{ada4}$, $t^{ada5}$: these parameters influence when the complex adaptation methods are triggered. They represent the absolute error of the soft sensor prediction that has to be exceeded in order to start the relevant adaptation. This, on one hand, influences the frequency with which the adaptation is performed and, on the other hand, theoretically defines the maximum prediction error of the model. Although at the moment, there is no formal mechanism that would guarantee the maximum error, it provides the possibility to implement such a mechanism and it will be the focus of further research. Having

such a mechanism, these parameters could be set according to the requirements for the accuracy of the predictions defined, for example, by the process control system. In the special case when the three parameters are set to the same value, i.e $t^{ada3} = t^{ada4} = t^{ada5}$, it leads to the adaptation approach discussed in Section 6.2.2. In general, this will be the case, unless stated otherwise, in the experiments presented in this chapter.

One of the benefits of the soft sensing algorithm which should be highlighted is that it is provided with recommendations for the parameter settings in form of the default values. Although the performance of the soft sensors developed using these values may potentially be sub-optimal, it provides an useful starting point in a situation when no expert knowledge, which could be used for more optimal parameter selection, is available. This aspect of the algorithm is subject to the experiments presented in Section 6.6.4, where soft sensors for different processes are developed using the default parameter settings.

In summary, this section shows that, although the proposed soft sensing algorithm consists of a large number of parameters, there are only a few parameters that require the attention of the soft sensor developer and have to be tuned for each separate task. Furthermore, the tuning of the critical parameters is straight-forward and intuitive as these are set according to the desired prediction accuracy.

## 6.5 Adaptation mechanisms summary

The adaptation mechanisms of the soft sensing algorithm defined throughout Section 6.2 play a prominent role for the resulting models and are therefore summarised in this section.

In the context of the architecture, the adaptation mechanisms can be categorised according to the level they operate at. The different adaptation loops and their role in the architecture are shown in Figure 6.10.

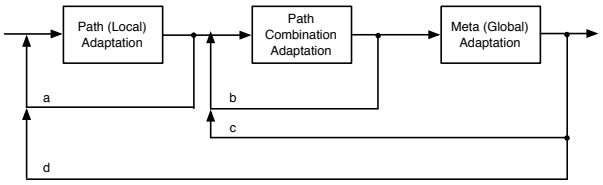



Figure 6.10: Adaptation loops provided within the architecture

Table 6.4 presents a summary of the adaptation mechanisms of the complex soft sensing algorithm together with their role in the architecture.

## 6.6 Experiments - soft sensors

This section is the final step on the pathway for soft sensor development in Figure 1.3. By applying the soft sensing algorithm from this chapter to the three process industry data sets used in Chapter 4, soft sensors for the industrial drier, thermal oxidiser processes and catalyst activity prediction are built and analysed. This soft sensor is in the following sections referred to as Robust On-line Soft Sensor (ROSS).

The data handling is done equivalently to the experiments in Chapter 4, see Section 4.4.3 for details.

| Adaptation Type | Loop | Description of the mechanism |
|---|---|---|
| 1 | a | Adaptation of the computational paths, i.e. local experts, this adaptation type is available only for the LWPR-based computational paths |
| 2 | b | Adaptation of the local expert descriptors $\mathcal{L}$ |
| 3 | c | Selection of the currently best performing path combination |
| 4 | c | Adaptation, i.e. rebuilding, of the path combinations |
| 5 | d | Deployment of new receptive fields, i.e. building new local experts, pruning of the local expert population, etc. |

Table 6.4: Overview of the different adaptation mechanisms

Concerning the set-up of the complex soft sensing algorithm, in all experiments the basis is the default parameters shown in Table 6.1 and only parameters that differ from these settings will be mentioned for each experiment. With respect to the analysis of the adaptive behaviour of the algorithm there are two different cases studied in the experiments. First, a case where $100\%$ of the target values are available for the adaptation. This is equal to the scenario considered in Chapter 4 and targets mainly the exploitation and analysis of the adaptation capabilities of the algorithm. This is followed by a more realistic case, where only $25\%$ of the target values are available for adaptation. In this case, all of the target values are still used for the calculation of the MSE and correlation coefficient measures of the soft sensors. This allows one to compare the results between the two scenarios.

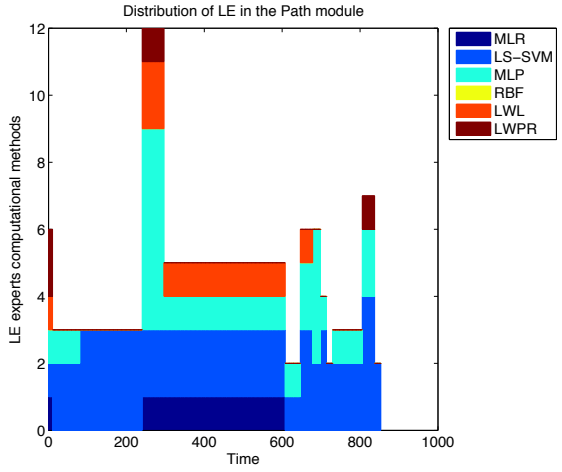
The subsequent experiments deal with different aspects of the soft sensing algorithm, including:

- the analysis of the influence of the population size in Section 6.6.1
- the analysis of the influence of the amount of available target data during the on-line phase in Section 6.6.2
- the ability to deploy low complexity soft sensors in an environment with limited resources in Section 6.6.3
- the transferability of the soft sensing algorithm, i.e. the performance of soft sensors developed using the default settings from Table 6.1 in Section 6.6.4
- the ability to deploy soft sensors with minimal amount of training data in Section 6.6.5

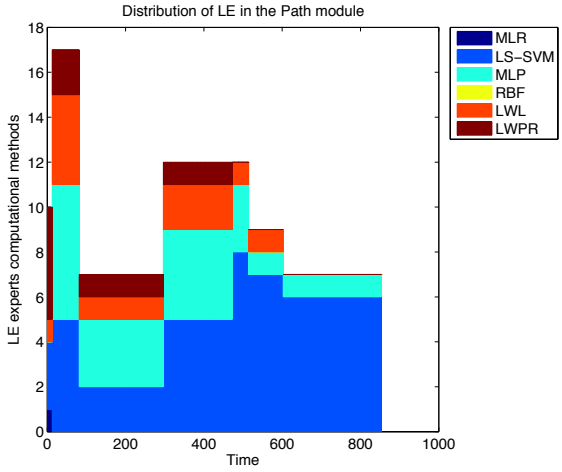
### 6.6.1 Analysis of the population size influence on the performance of the soft sensor

The aim of these experiments is to analyse the influence of the target size of the local expert population $n^{LE,target}$ on the performance of the soft sensors. This analysis is of interest because this parameter belongs to the group of parameters that can potentially have significant influence on the performance of the soft sensor and should therefore be considered for each task separately. However, due to the flexibility of the proposed soft sensing algorithm, a certain level of robustness of the algorithm with respect to the $n^{LE,target}$ parameter value is expected.

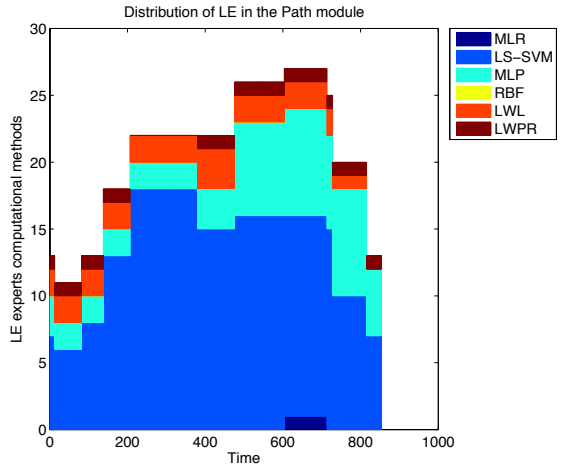
In order to prove the functional capability of the population size control mechanism, Figure 6.11 shows the size and composition of the local experts in the Paths module for different settings of target value population size for the industrial drier data set. The figure shows that the simple approach for the control of the population size is able to maintain a stable population size that is close to the desired number of local experts $n^{LE,target}$. The step changes in the plots correspond



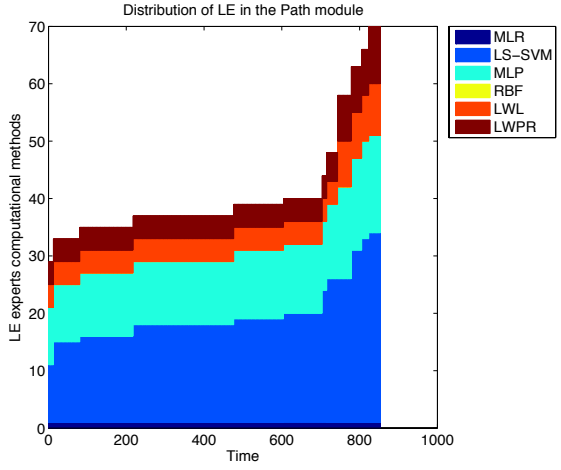

(a) $n^{LE,target} = 1$



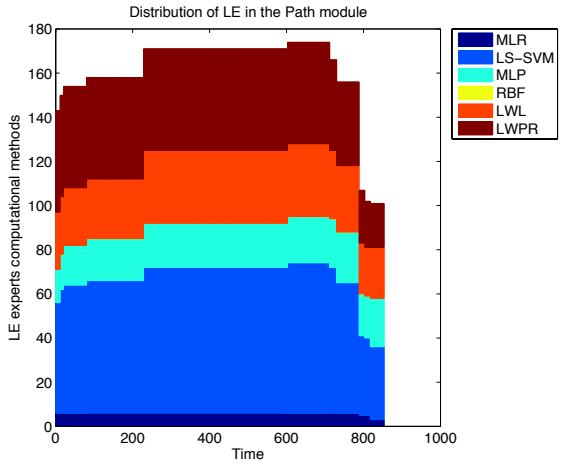

(b) $n^{LE,target} = 10$




(c) $n^{LE,target} = 20$




(d) $n^{LE,target} = 50$




(e) $n^{LE,target} = 100$

Figure 6.11: The effect of changes of the parameter controlling the desired size of the local expert population (industrial drier data set)

to Type 5 adaptations where new local experts are launched. The purpose of the population size control is not only to limit the computational resources but also to implement an effective *pruning* method where the local experts are pruned based on their mutual correlation (see the diversity selection mechanism in Section 6.2.1 for details). Using this mechanism a stable and diverse population of local experts is effectively maintained.

### Industrial drier

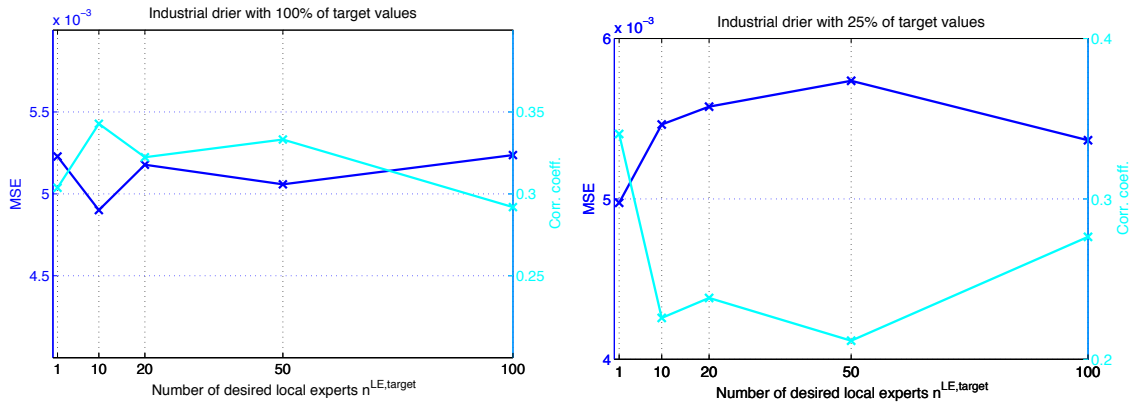
For this data set, the parameter(s) that differ(s) from the default settings is/are:

- $t^{ada3} = t^{ada4} = t^{ada5} = 0.15$

The performance of soft sensors with different desired population sizes for the industrial drier data set is presented in Figure 6.12 by the means of yy-plot. The plot accommodates two y-axes in the same plot. The left-hand side axis shows the MSE, whereas the right-hand side is the axis for the correlation coefficient.

As a general pattern, one can observe that the performance, in terms of the MSE as well as the correlation coefficient, is not very sensitive with respect to the setting of the $n^{LE,target}$ parameter. For example, for the case shown in Figure 6.12(a) the difference between the extreme values of the parameter is for squared error measure less than $10\%$.

Furthermore, it can be observed that despite the fact that only $25\%$ of the target values were used for the adaptation in Figure 6.12(b), the performance is similar to the case where $100\%$ of target values were available. This confirms the conclusion that was made in Chapter 4 where it was found that for this data there is only a limited amount of adaptation required. This effect will be analysed in more detail in Section 6.6.2 where the influence of the amount of available target data is studied.




(a) $100\%$ of target values for adaptation

(b) $25\%$ of target values for adaptation

Figure 6.12: Industrial drier: The MSE and correlation coefficients achieved with different population sizes

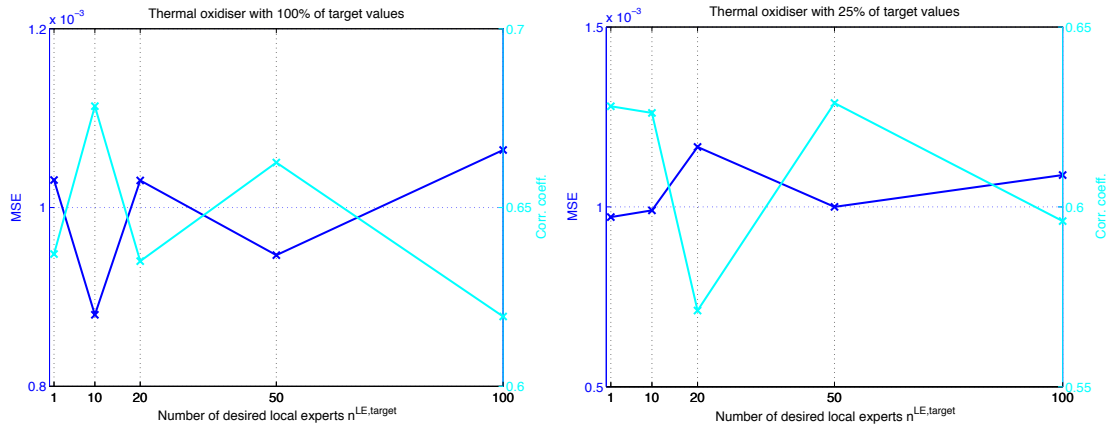
### Thermal oxidiser

For this data set, the parameter(s) that differ(s) from the default settings is/are:

- $t^{ada1} = t^{ada2} = t^{ada3} = 0.08$

For the thermal oxidiser there is no obvious pattern between the performance and the $n^{LE,target}$ parameter as shown in Figure 6.13. This demonstrates the robustness of the algorithm with respect to the value of the analysed parameter. Nevertheless, there is a slight decrease in the performance, i.e. increase in MSE and decrease in correlation coefficient, with increasing population size. The performance sensitivity for this data set is similar to the previous data set.

One can also observe a similar performance between the two considered scenarios shown in Figure 6.13. This effect will be further analysed in Section 6.6.2.




(a) 100% of target values for adaptation

(b) 25% of target values for adaptation

Figure 6.13: Thermal oxidiser: The MSE and correlation coefficients achieved with different population sizes

### Catalyst activation

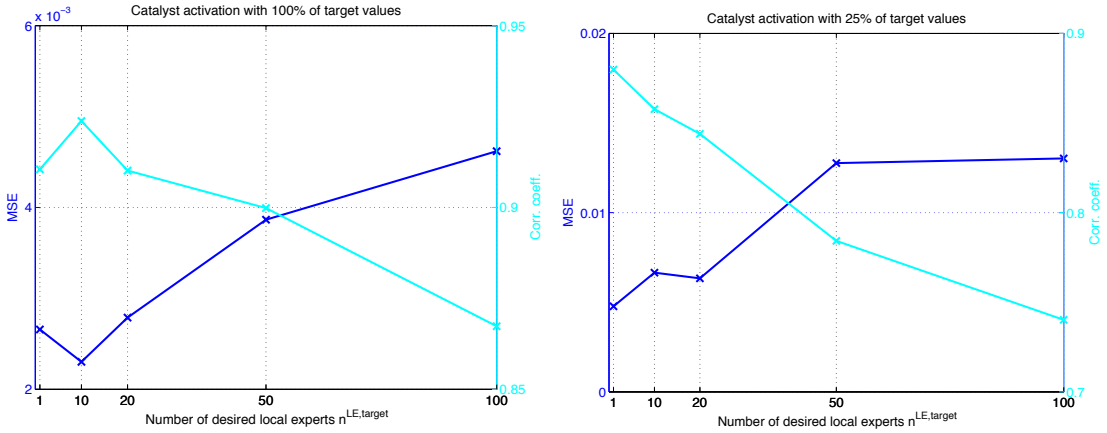
For this data set, the parameter(s) that differ(s) from the default settings is/are:

- $t^{ada3} = t^{ada4} = t^{ada5} = 0.08$

In the case of this data set, a steadily decreasing performance with increasing population size can be observed in Figure 6.14. In Chapter 4 it was shown that this data set requires frequent adaptation in order to obtain useful predictions. This fact is also reflected in the results presented here because with growing population size there is an increasing amount of *old* out-of-date local experts in the population. These local experts in turn *disturb* the more accurate predictions of newer models and affect the overall performance of the soft sensor. The sensitivity of the results also appears to be higher compared to the previous data sets and therefore the selection of the population size should be considered carefully.

### Experiment conclusion

The observations made during these experiments vary from data set to data set, which indicates that this parameter should be carefully considered for each task. Nevertheless, for two of the three considered data sets the sensitivity of the performance with respect to the analysed parameter was rather low and the setting $n^{LE,target} = 10$ can be considered in terms of the presented results as an optimal default value for this parameter.




(a) 100% of target values for adaptation

(b) 25% of target values for adaptation

Figure 6.14: Catalyst activation: The MSE and correlation coefficients achieved with different population sizes

Another general observation is that the population size is related to the dynamics of the underlying data. In particular, it appears that for dynamic data sets larger population sizes lead to lower performance. The intuitive explanation of this effect is that highly dynamic data set do not only require frequent adaptation but also benefit from a small population of local experts focused on the latest data.

### 6.6.2 Influence of the availability of target data

These experiments target the analysis of the influence of the amount of available target data. In the experiments in Chapter 4, two cases were examined: (i) non-adaptive scenario where no target data were available and (ii) fully adaptive scenario where, after making the prediction, a correct target value was provided for each on-line data point. In this section, these two extreme cases, i.e. either $0\%$ or $100\%$ of target values, are extended with several more cases. To be able to study the influence of the availability of the target data on the performance of the soft sensors, there are cases with $0\%$, $25\%$, $50\%$, $75\%$ and $100\%$ of available target values considered. Intuitively, it can be expected that the performance of the soft sensor will increase with increasing percentage of target data allowing more frequent adaptation.

In order to be able to assess the performance of the soft sensors developed by means of the complex soft sensing algorithm, their performance will be compared to the optimally parametrised LWPR-based soft sensors (see Sections 4.4.4 - 4.4.6 for details of the LWPR-based soft sensors).

#### Industrial drier

For this data set, the parameter(s) that differ(s) from the default settings is/are:

- $t^{ada3} = t^{ada4} = t^{ada5} = 0.15$

The results of this experiment are shown in the yy-plot in Figure 6.15. The figure compares the performance of ROSS to the LWPR-based soft sensors applied under equivalent conditions. For this data set, the plain LWPR delivers better performance than the complex soft sensing algorithm, which confirms the strong performance of the LWPR algorithm also found in Section 4.4.4. The reason for the good performance is probably its ability to deal with the high noise level of the target variable.

Another interesting fact is that for both model types, the performance remains stable for the cases, where the amount of available target data is larger than $50\%$. In other words, it means that in order to operate the soft sensor only half of the currently collected target values are required and the rest is, from the point of view of the soft sensor, redundant. This represents an ideal environment for the application of an adaptive soft sensor because on one hand the soft sensor can deliver accurate predictions and on the other hand requires only $50\%$ of the target values for its adaptation allowing one to skip the remaining $50\%$ of the (costly) measurements. If applied, such a soft sensor could lead to a significant reduction in the manually collected target values.

#### Thermal oxidiser

For this data set, the default values are used for all parameters.

The results are shown in Figure 6.16. The figure again compares the performance of the LWPR-based soft sensor to the performance of ROSS for the different settings of the available target values percentages. As one can see the performance of ROSS is better for all settings.

This data set is another case where a large part of the target value collection can be omitted. As shown in Figure 6.16, the performance remains almost constant if $25\%$ or more of the target values is provided for adaptation which means that to adapt the soft sensor and keep an acceptable performance level only every fourth target sample needs to provided.

In order to illustrate the above fact, Figures 6.17(a) and 6.17(b) show the predictions of the soft sensor using $25\%$ and $100\%$ available target values respecitvely. The figure shows that the predictions are very similar and in both cases follow the target values well.

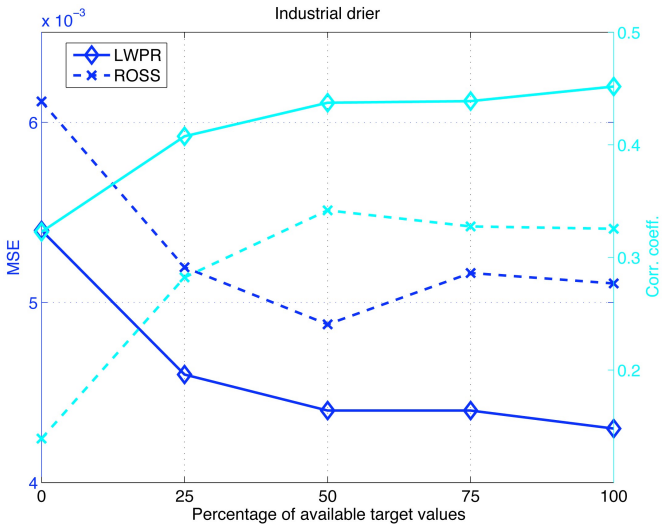



Figure 6.15: Industrial drier: The effects of changing amount of target values, comparison between the LWPR-based soft sensor and ROSS

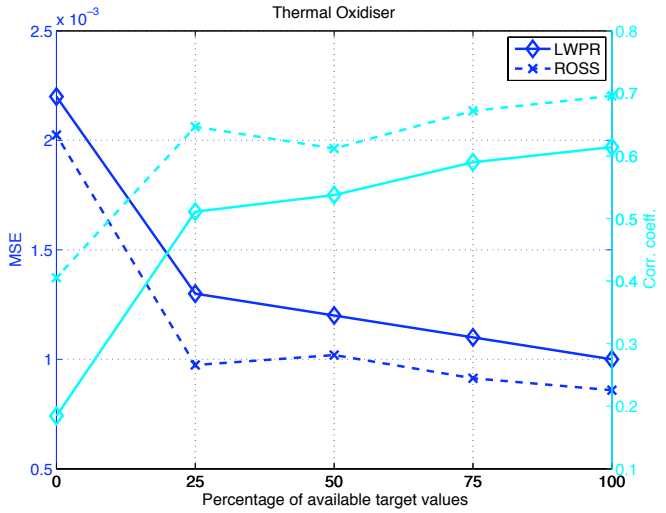


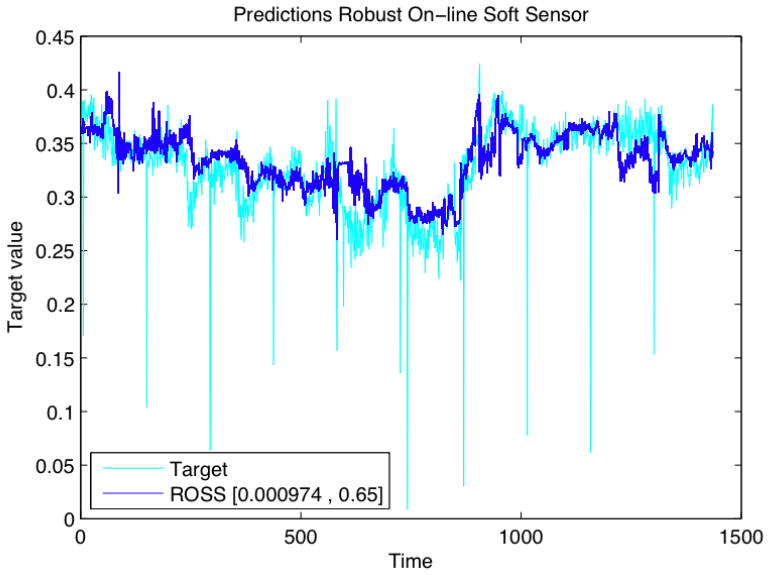

Figure 6.16: Thermal oxidiser: The effects of changing amount of target values, comparison between the LWPR-based soft sensor and ROSS

### Catalyst activation

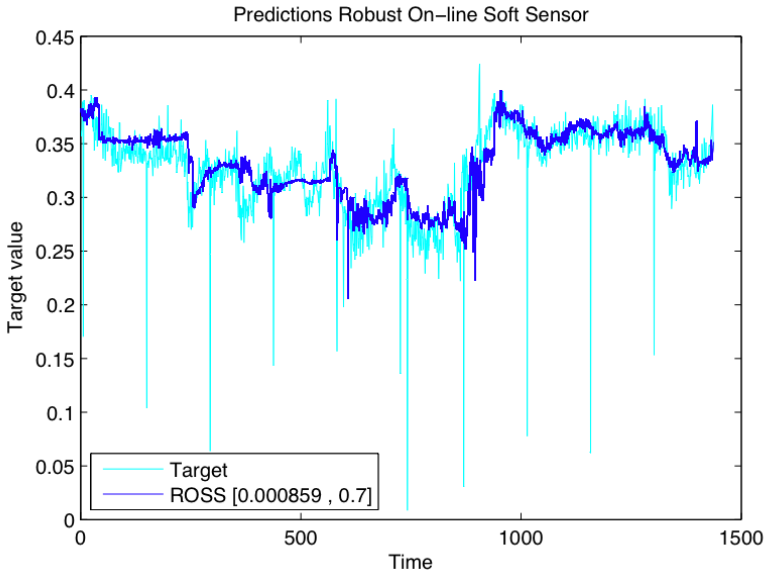
For this data set, the parameter values do not differ from the default parameter set from Table 6.1.

Figure 6.18 shows the performance plot and compares the LWPR-based soft sensor with ROSS. Please note, in this figure a logarithmic scale had to be used for the left-hand side y-axis (MSE). The reason for this is that the performance of the non-adaptive soft sensor is significantly lower than that of the adaptive models (see Section 4.4.6 for discussion on this topic). It can be observed that, similarly to the previous data set, for most settings the ROSS outperforms the LWPR-based soft sensor.

The same figure also shows that the performance of the soft sensors strongly improves with increasing availability of the target values. This observation is consistent with the findings from




(a) 25% of target values for adaptation




(b) 100% of target values for adaptation

Figure 6.17: Thermal oxidiser: Comparison between ROSS predictions using 25% and 100% of the available target values for adaptation

Chapter 4. Unlike in the previous two cases, in order to achieve acceptable performance of the soft sensor, there are as many target data as possible required. Therefor, the role of the soft sensor can, for example, be to deliver real-time prediction of the target variables and in this way bridge the delays related to the physical measurement of the catalyst activity in laboratories.

The low performance of the non-adaptive soft sensors is demonstrated in Figure 6.19. It is clearly visible that neither of the non-adaptive soft sensor is able to follow the changes of the target variable and that both soft sensors continue to predict the same values for the whole duration of

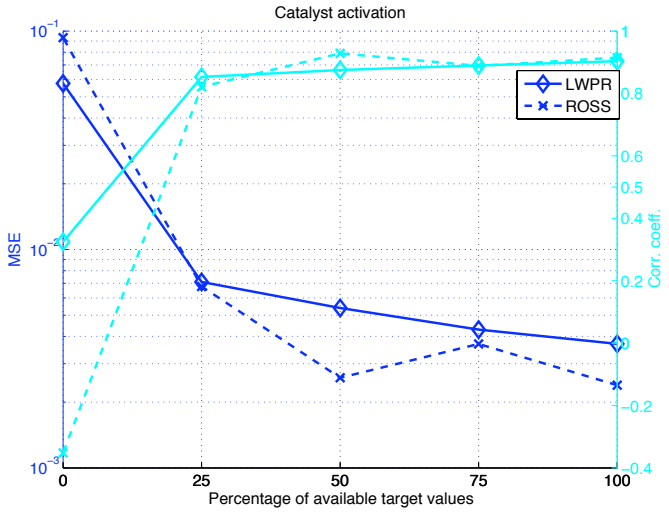


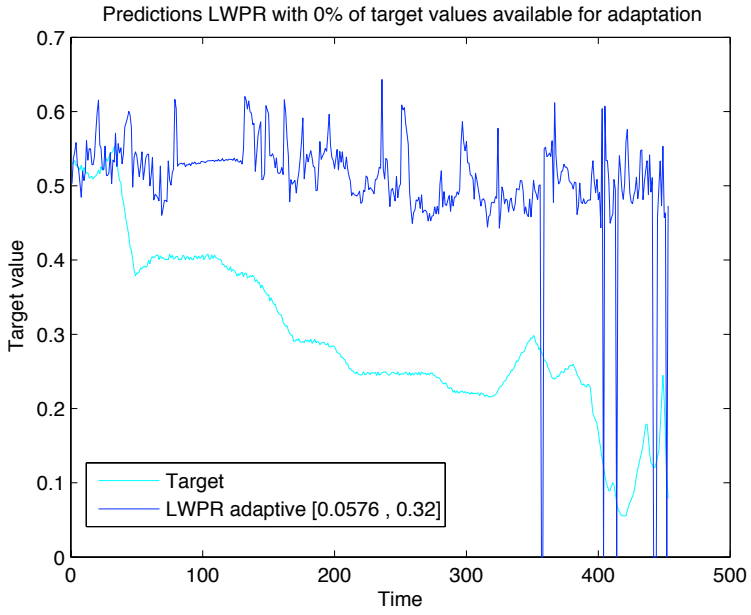

Figure 6.18: Catalyst activation: The effects of changing amount of target values, comparison between the LWPR-based soft sensor and ROSS (note the logarithmic left-hand side y-scale)

the on-line phase. In such a scenario an adaptive mechanism for the maintenance of the soft sensor is inevitable.

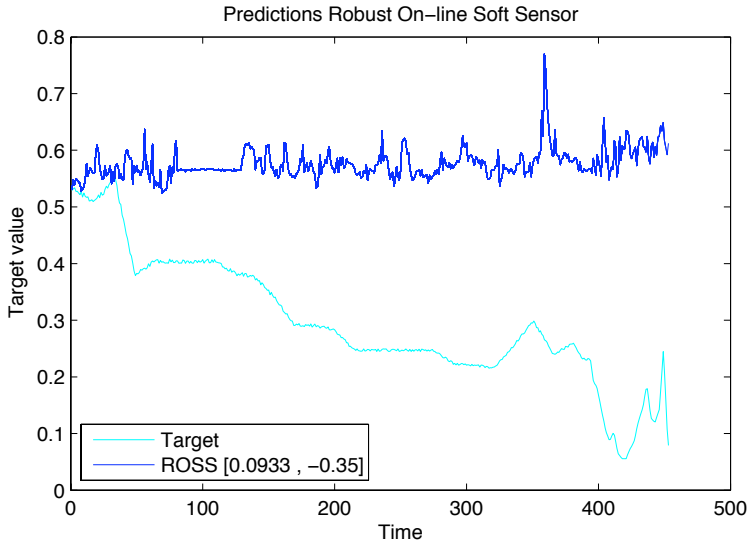
The prediction performance improves dramatically once the feedback information is used for the adaptation, as demonstrated in Figure 6.20, which shows the same soft sensors as above, but this time using $50\%$ of the target values for adaptation.

### Experiment conclusion

These experiments revealed two cases where adaptive models can provide not only useful on-line predictions of the target values but can also lead to a strong reduction of the amount of collected target values, which in turn could potentially have significant financial implications on the process. For the third data set (catalyst activation), the soft sensor requires as many of the target values for its adaptation as possible.

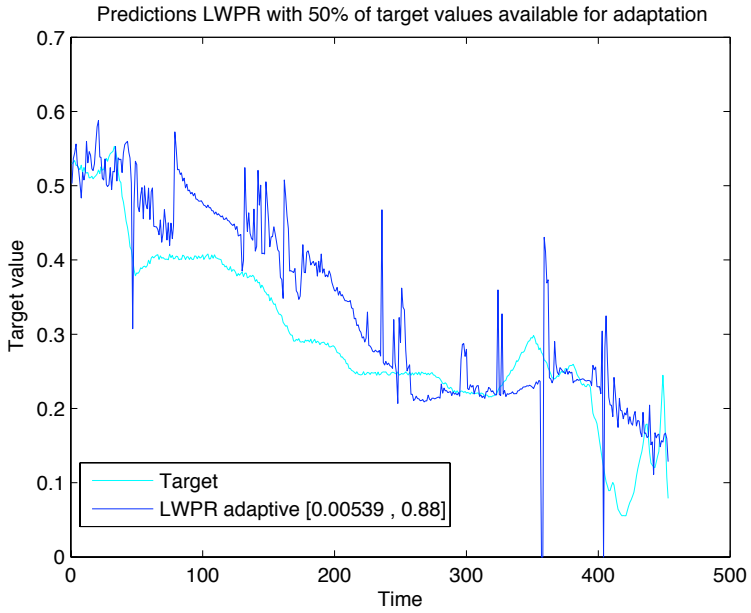



(a) Non-adaptive LWPR-based soft sensor

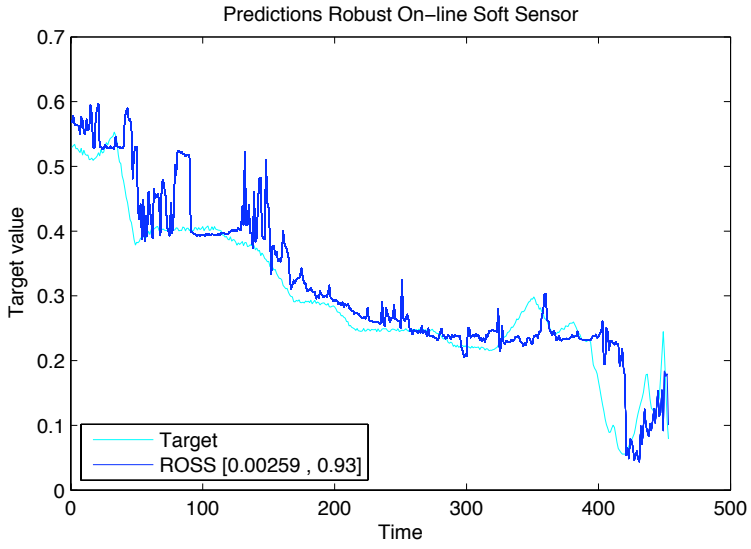



(b) Non-adaptive ROSS

Figure 6.19: Catalyst activation: Low performance of the non-adaptive soft sensors




(a) Adaptive LWPR-based soft sensor




(b) Adaptive ROSS

Figure 6.20: Catalyst activation: Predictions of two adaptive soft sensors using $50\%$ of target values for adaptation

### 6.6.3 Low complexity soft sensors

These experiments focus on two potential issues of the proposed soft sensing algorithm, namely its high complexity and high number of input parameters. The latter topic was theoretically discussed in Section 6.4 and will be dealt with in an empirical way here. The aim of the subsequent experiments is to demonstrate that the complexity of the proposed soft sensing algorithm can be exploited to develop simple, i.e. low complexity, soft sensors that keep an acceptable level of performance. At the same time, the input parameters for all of the soft sensors are set to the default values from Table 6.1, which aims at the analysis of the algorithm's capability to deploy a soft sensor without any manual parameter optimisation. This latter aspect will also be further studied in Section 6.6.4, where a more complex version of the soft sensors is developed under similar contraints.

In order to limit the complexity of the resulting soft sensor, the pools of the pre-processing and computational learning methods were limited to the averaging filters, robust PCA and the multilinear regression predictors, i.e. the PPMP and CLMP consist (see Figure 5.5) of the following methods only:

- SF: $n^{smooth} = [1, 4, 7, 10]$
- RobPCA: $t^{covVar} = 0.95$
- MLR.

Additionally, the population size $n^{LE,target}$ and the number of LEC candidates trained per receptive fields $n^{LEC}$ were limited to:

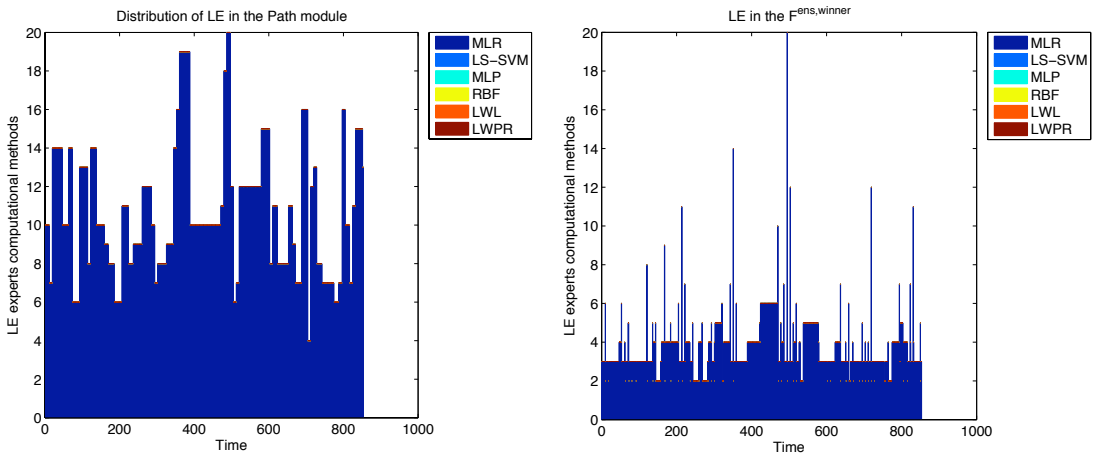
- $n^{LE,target} = 10$
- $n^{LEC} = 10$.

#### Industrial drier

The first plot, which is considered here is the size and composition of the local expert population in Figure 6.21(a). It shows that the population size fluctuates at around ten, which is the desired population size. Furthermore, the plot shows that there are only local experts using the MLR predictor, which also corresponds to the settings of the algorithm. Another interesting aspect is revealed in Figure 6.21(b), which shows the local experts in the $\mathcal{F}^{ens,winner}$ ensemble, which is the actual model making the final predictions. The figure shows that the ensemble size is quite small, most of the time it is between two and four local experts. This in turn indicates that very simple (linear combinations of two to six linear models) locally valid models are responsible for the predictions. An additional benefit of such a simple model is its transparency. Since the models consist merely of PCA pre-processing and linear combination of the variables, it is possible to extract, for example, the contribution of the particular variable to the predictions and other useful information about the models and the predictions.

One can also see several *spikes* in Figure 6.21(b). These correspond to unsuccessful Type 3 and Type 4 adaptations. These adaptation mechanisms re-select the winning ensemble $\mathcal{F}^{ens,winner}$ and re-build the ensembles respectively (for details see Section 6.5). One can see that the adaptation was unsuccessful because the number of local experts in the winning ensemble changes immediately after the adaptation, which leads to the spikes.

The final prediction, as well as the MSE and correlation coefficient values, can be found in Figure 6.22. The performance of this simple soft sensor is quite high, in fact it is comparable to




(a) local experts population (b) compositions of the winning ensemble $\mathcal{F}^{ens,winner}$

Figure 6.21: Industrial drier: Local experts population and the composition of the winning ensemble $\mathcal{F}^{ens,winner}$ (100% of target values for adaptation)

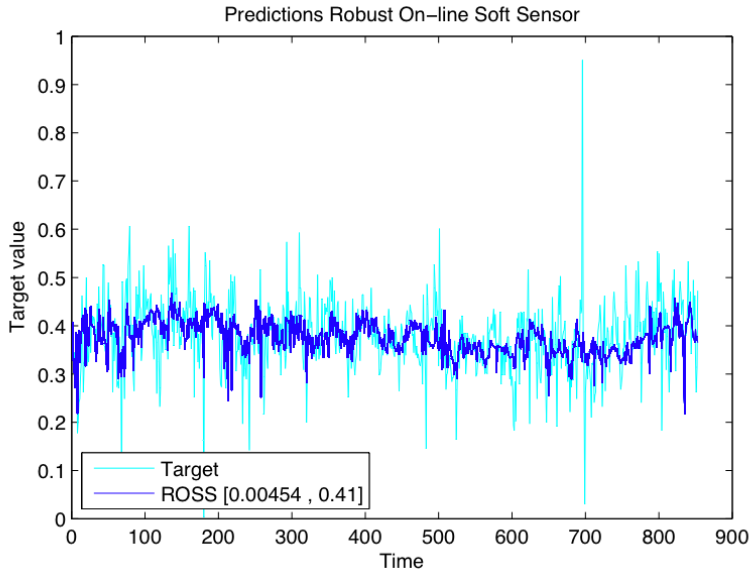
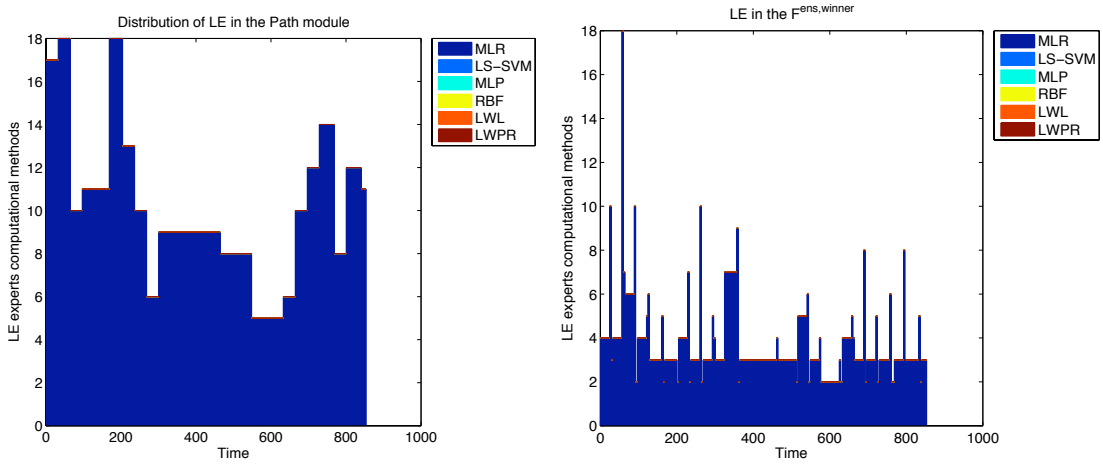



Figure 6.22: Industrial drier: Predictions of the minimal soft sensor (100% of target values for adaptation)

the results achieved by the LWPR and LASSA soft sensors in Chapter 4, see Section 4.4.4 for comparison.

In the case where only 25% of the targets are available for adaptation, the figures look similar with the only obvious difference being that the adaptation is less frequent, see Figure 6.23 for the local expert plots and Figure 6.24 for the predictions. Although it is obvious that the prediction performance is impacted by the less frequent target values, the soft sensor is still able to deliver useful predictions.




(a) local experts population (b) compositions of the winning ensemble $\mathcal{F}^{ens,winner}$

Figure 6.23: Industrial drier: Local experts population and the composition of the winning ensemble $\mathcal{F}^{ens,winner}$ (25% of target values for adaptation)

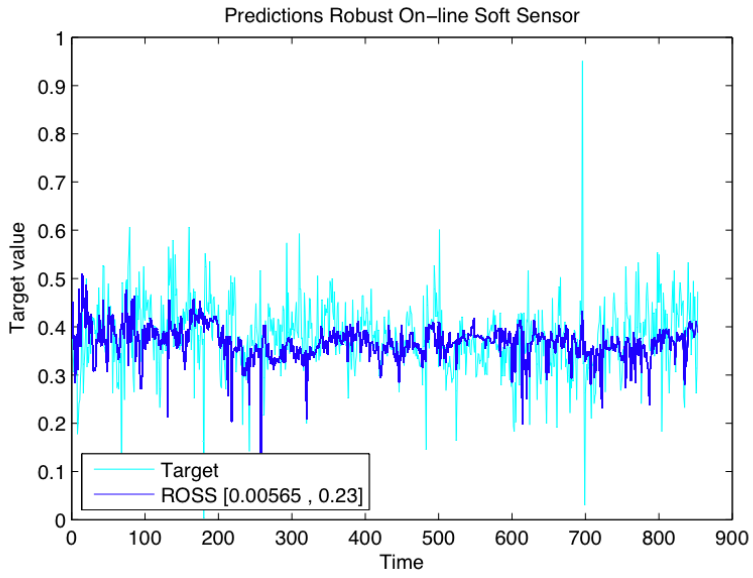


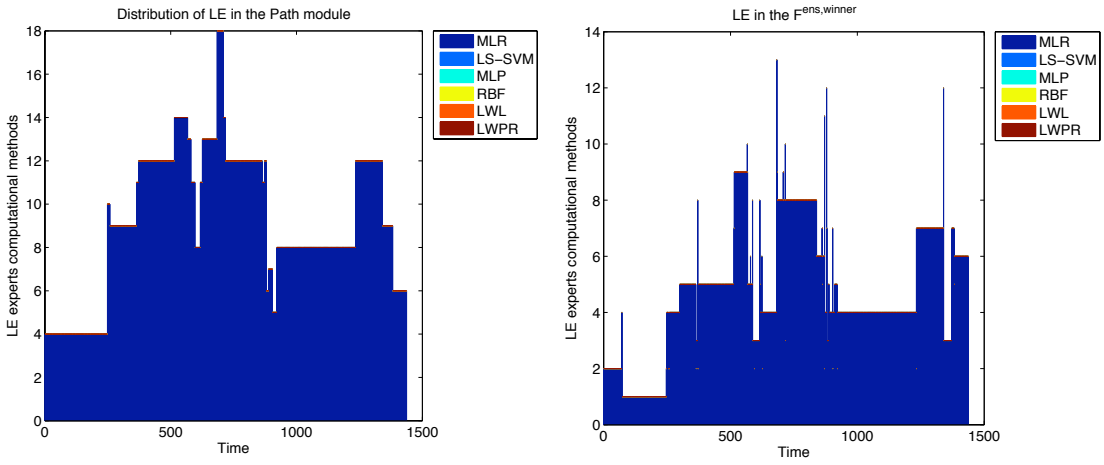

Figure 6.24: Industrial drier: Predictions of the minimal soft sensor (25% of target values for adaptation)

### Thermal oxidiser

For this data set, the local expert population is again fluctuating around the desired ten local experts as shown in Figure 6.25(a). Figure 6.25(b) reveals some interesting aspects of the model. Although the parameter settings for the previous and the current data set are equal, one can observe that the size of the winning ensemble is on average higher for this data set than it was for the previous data set (compare Figure 6.25(b) with Figure 6.21(b)). Another interesting fact that can be observed from the same figure is that, for the range beginning with time sample 73 and ending with sample 246, there is only a single local expert in the winning ensemble. This demonstrates the flexibility

of the applied ensemble building mechanism, i.e. the co-operative selection presented in Section 6.2.1.

The final predictions of the soft sensor are following the target values well (see Figure 6.26). In fact, the performance of this simple soft sensor is again similar to the LWPR soft sensor presented in Section 4.4.5.




(a) local experts population (b) compositions of the winning ensemble $\mathcal{F}^{\rceil\backslash f,\sqsupseteq\rangle\backslash\backslash\rceil\nabla}$

Figure 6.25: Thermal oxidiser: Local experts population and the composition of the winning ensemble (100% of target values for adaptation)

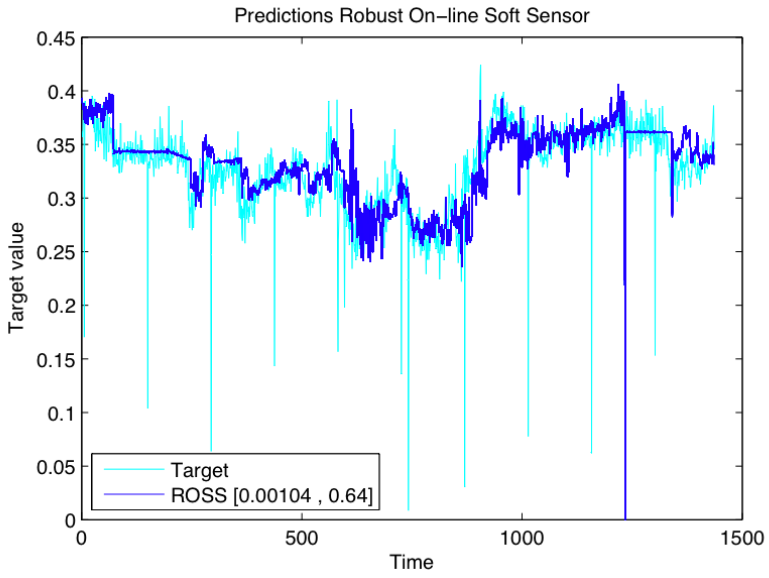



Figure 6.26: Thermal oxidiser: Predictions of the minimal soft sensor (100% of target values for adaptation)

In the case of a limited amount of target data, the performance of the simple soft sensor remains surprisingly good as demonstrated in Figure 6.27. The figure shows that the soft sensor is still able to follow the general trend of the data, i.e. switching between the two states of the target value,

but it fails to react to the smaller changes in the target value.

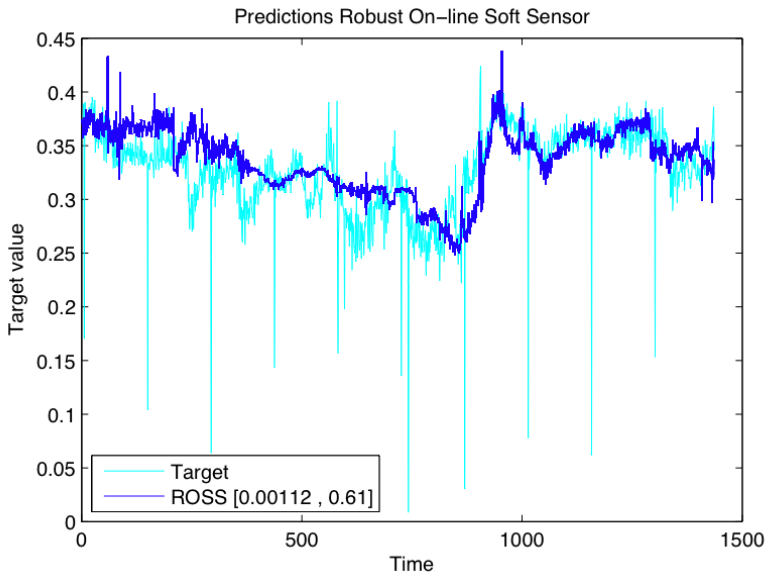



Figure 6.27: Thermal oxidiser: Predictions of the minimal soft sensor (25% of target values for adaptation)

### Catalyst activation

The final prediction, the MSE and the correlation coefficient can be found in Figures 6.28 and 6.29. Although the simple soft sensor does not achieve the performance of the full-scale soft sensors presented in Section 6.6.2, its performance is again on the same level as the LWPR-based soft sensor (see Section 4.4.5).

The experiments with a low number of target data show a strong decrease in the performance of the soft sensor as expected. Figure 6.6.3 shows that although the soft sensor is able to follow the general trend of the data, there is an offset between the predictions and correct target values, an effect that was also present with the other soft sensor types (see Section 6.6.2).

.

### Experiment conclusion

The experiments with the low complexity soft sensor presented in this section have shown that, despite the limited resources, the developed soft sensors were able to deliver useful predictions. Interestingly, the performance of these soft sensors was at a similar level as the performance of the corresponding LWPR-based soft sensors presented in Section 6.6.2. Taking into account that the model delivering the output prediction was a combination of a low number of linear models this is a very good result, which demonstrates how the complex mechanisms which are part of the model can help to deliver simple and efficient soft sensors.

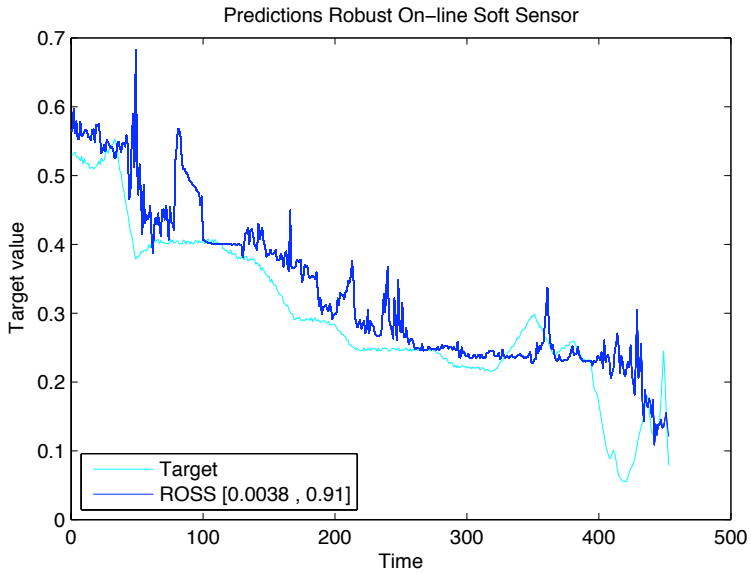



Figure 6.28: Catalyst activation: Predictions of the minimal soft sensor ($100\%$ of target values for adaptation)

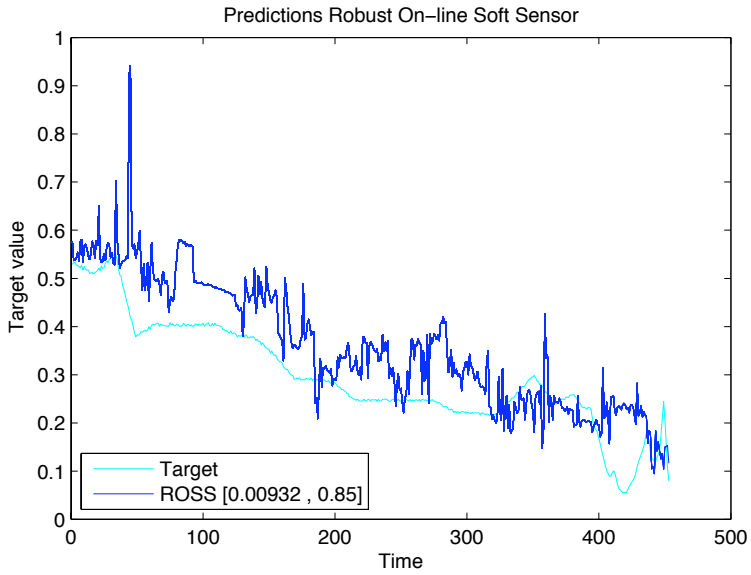
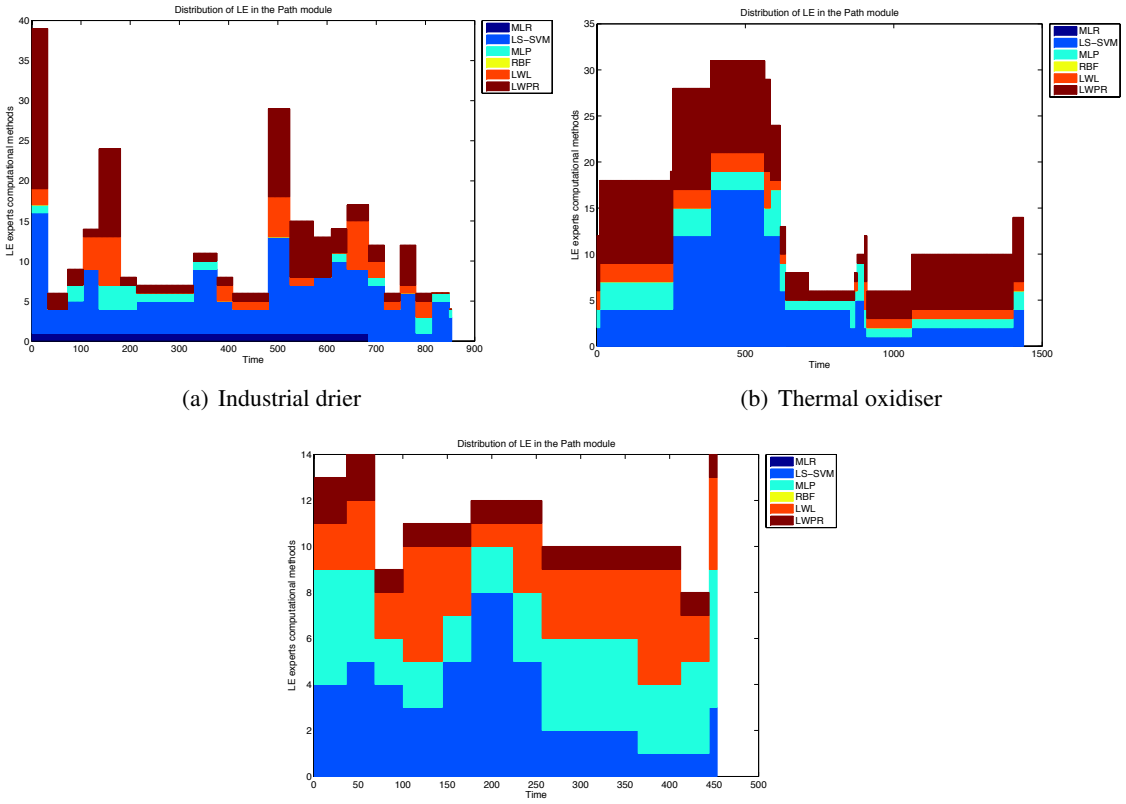



Figure 6.29: Catalyst activation: Predictions of the minimal soft sensor ($25\%$ of target values for adaptation)

### 6.6.4 Transferability experiments

The aim of these experiments is to analyse the ability of the soft sensing algorithms to change the structure of the developed model according to the task that has to be dealt with. For this reason, the algorithm with exactly the same parameter settings is applied to the three data sets used throughout this work. The resulting soft sensors will be analysed with respect of their internal structure, i.e. the composition of the local experts and the achieved performance. The parameter settings of the algorithm for all three data sets are consistent with the default values presented in Table 6.1.

The first indication of the adaptation of the soft sensing algorithm to the different tasks is presented in Figure 6.30. The figure shows the compositions of the local expert population for




(a) Industrial drier

(b) Thermal oxidiser

(c) Catalyst activation

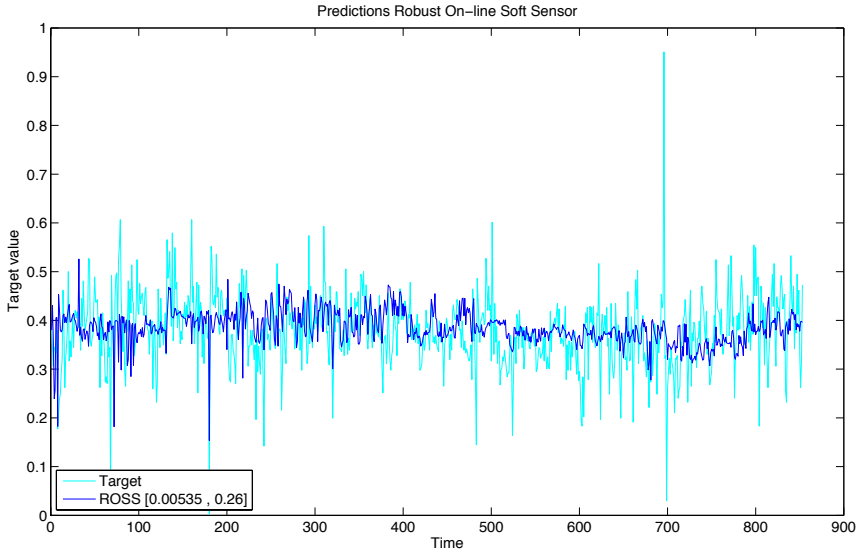
Figure 6.30: Composition of the local experts population for the three data sets (25% of target values for adaptation)

the three data sets. In terms of the population size, it fluctuates around the desired value of the population size of ten local experts for all three data sets. As for the composition of the predictive techniques used for the local experts, there are large differences. This is evidence that the algorithm is able to match the selected techniques with the underlying task. The pool of local experts for the drier data set, see Figure 6.30(a), has the most diverse and changing composition dominated mainly by the LSSVM and LWPR techniques. It is also the only case where the linear regression (MLR) is represented in the local expert population. In the case of the thermal oxidiser data set (Figure 6.30(b)), there is an overwhelming domination by the LSSVM and LWPR local experts throughout the entire on-line phase. In contrast to this, for the catalyst activation task (Figure 6.30(c)), there is a certain preference for the MLP and LWL methods.

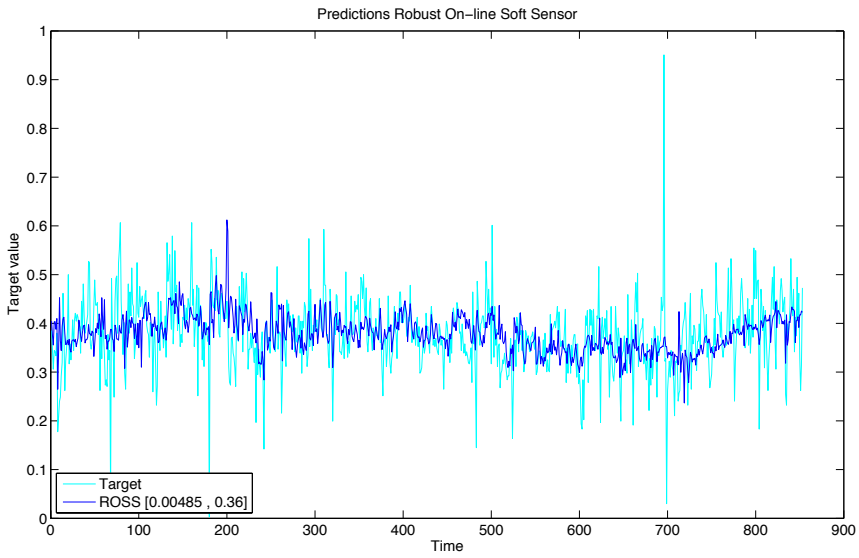
The following sections deal with predictions and performances of the three soft sensors developed using the complex soft sensing algorithm. These soft sensors are referred to as *full-scale* soft sensors.

### Industrial drier

The prediction for the two cases, where either $25\%$ or $100\%$ of target data are available, are presented in Figure 6.31.




(a) $25\%$ of target values for adaptation




(b) $100\%$ of target values for adaptation

Figure 6.31: Industrial drier: Predictions of the full-scale soft sensors

By the means of visual inspection, it is possible to conclude that the soft sensors are providing useful predictions. The performance of the two full-scale soft sensors is compared to the low complexity soft sensors discussed in Section 6.6.3, and to the LWPR-based soft sensor discussed

in 6.6.2, in Table 6.5.

| Target value availability | Soft sensor type | MSE | Corr. coef. |
|---|---|---|---|
| 25% | LWPR | $\mathbf{4.57 * 10^{-3}}$ | **0.41** |
| | Minimal ROSS | $5.65 * 10^{-3}$ | 0.23 |
| | Full-scale ROSS | $5.35 * 10^{-3}$ | 0.26 |
| 100% | LWPR | $\mathbf{4.33 * 10^{-3}}$ | **0.45** |
| | Minimal ROSS | $4.54 * 10^{-3}$ | 0.41 |
| | Full-scale ROSS | $4.85 * 10^{-3}$ | 0.36 |

Table 6.5: Industrial drier: Comparison between the LWPR-based soft sensor, minimal version of ROSS and full-scale ROSS

The table shows that the best performance for this data set is achieved by the LWPR-based soft sensor. Although the difference between the LWPR-based soft sensor and ROSS seems to be quite large, Figure 6.32 reveals that the predictions are very similar and that the difference in the MSE and correlation coefficient performance can be attributed mainly to following the peaks of the target variable, which is performed better by the LWPR-based soft sensor.

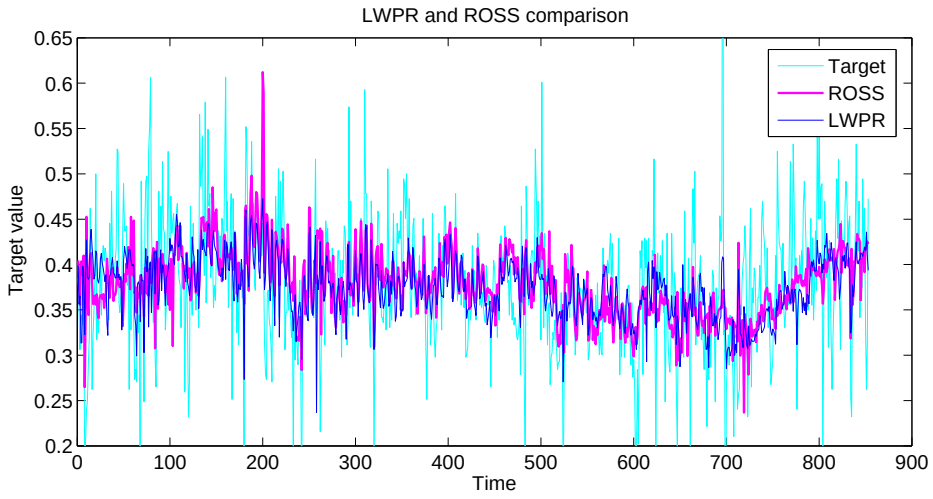
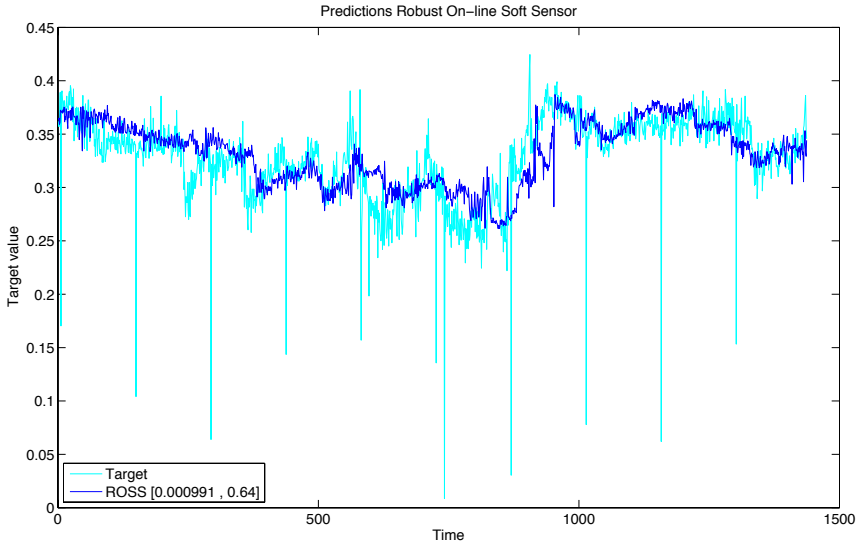



Figure 6.32: Industrial drier: Direct comparison between the LWPR-based soft sensor and ROSS

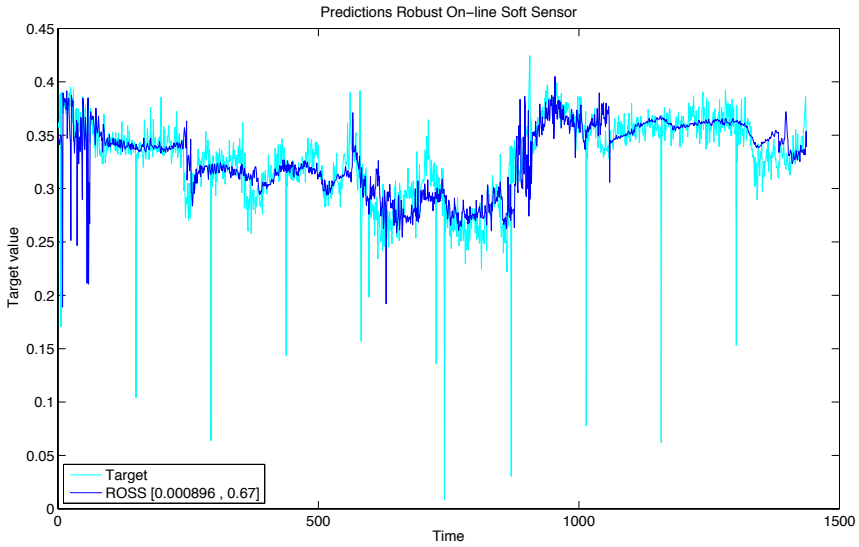
### Thermal oxidiser

The predictions of the two soft sensors for the thermal oxidiser data set are shown in Figure 6.33. It can also be observed that it is possible to develop a useful soft sensor for this data by applying the default parameter settings from Table 6.1.

The performance comparison of the thermal oxidiser shown in Table 6.6 shows a different pattern than the previous data set. Similarly to the experiments in Section 6.6.2, in this case a small difference between the 25% and 100% available target values can be observed. The full-scale ROSS delivered the best performance for both analysed cases. This shows that, in contrast to the drier data set, in this case it is beneficial to use the more complex soft sensor.




(a) 25% of target values for adaptation




(b) 100% of target values for adaptation

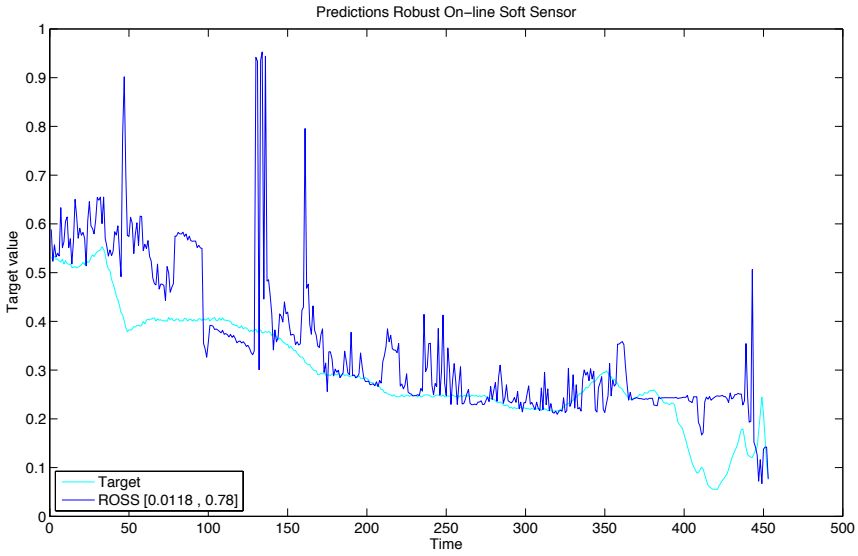
Figure 6.33: Thermal oxidiser: Predictions of the full-scale soft sensors

| Target value availability | Soft sensor type | MSE | Corr. coef. |
|---|---|---|---|
| 25% | LWPR | $1.27 * 10^{-3}$ | 0.51 |
| | Minimal ROSS | $1.12 * 10^{-3}$ | 0.61 |
| | Full-scale ROSS | $\mathbf{9.91 * 10^{-4}}$ | **0.64** |
| 100% | LWPR | $1.04 * 10^{-3}$ | 0.61 |
| | Minimal ROSS | $1.04 * 10^{-3}$ | 0.64 |
| | Full-scale ROSS | $\mathbf{8.96 * 10^{-4}}$ | **0.67** |

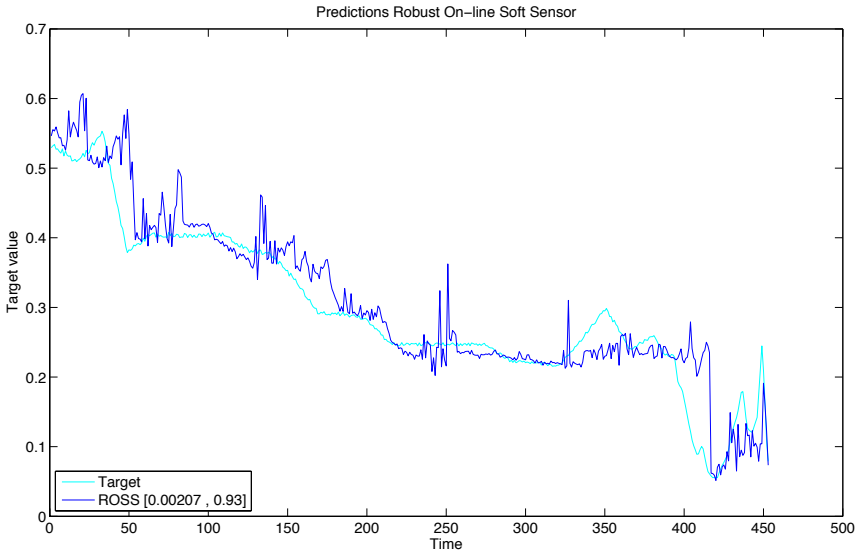
Table 6.6: Thermal oxidiser: Comparison between the LWPR-based soft sensor, minimal version of ROSS and full-scale ROSS

### Catalyst activation

Figure 6.34 shows the predictions of the full-scale ROSS for the catalyst activation data set. Again,




(a) 25% of target values for adaptation




(b) 100% of target values for adaptation

Figure 6.34: Catalyst activation: Predictions of the full-scale soft sensors

it can be confirmed that it was possible to develop a useful soft sensor by applying the complex soft sensing algorithm from Section 6.2 with the default parameter setting.

Table 6.7 compares the MSE and correlation coefficient of the full-scale soft sensor predictions to the previously presented LWPR-based soft sensor as well as the minimal version of ROSS. As for the case with 25% of available target values, the performance of both the minimal and the full-scale ROSS is lower than that of the LWPR-based soft sensors. This can be attributed mainly to the instability of the predictions around the samples 50,140 and 440, whereas for the other parts of the on-line data, full-scale ROSS delivers more accurate predictions shown in Figure 6.35.

| Target value availability | Soft sensor type | MSE | Corr. coef. |
|---|---|---|---|
| 25% | LWPR | $\mathbf{7.13 * 10^{-3}}$ | **0.85** |
| | Minimal ROSS | $9.32 * 10^{-3}$ | **0.85** |
| | Full-scale ROSS | $1.18 * 10^{-2}$ | 0.78 |
| 100% | LWPR | $3.72 * 10^{-3}$ | 0.90 |
| | Minimal ROSS | $3.80 * 10^{-3}$ | 0.91 |
| | Full-scale ROSS | $\mathbf{2.07 * 10^{-3}}$ | **0.93** |

Table 6.7: Catalyst activation: Comparison between the LWPR-based soft sensor, minimal version of ROSS and full-scale ROSS

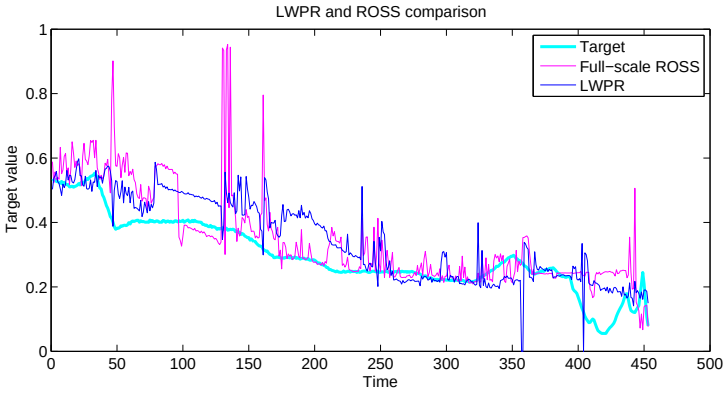



Figure 6.35: Catalyst activation: Direct comparison between the LWPR-based soft sensor and ROSS (25% of target data for adaptation)

In the case where 100% of the target data is available during the on-line phase, full-scale ROSS delivers the best performance by a large margin.

### Experiment conclusion

The experiments in this section have shown that the presented complex soft sensing algorithm has the capability to adjust the internal model structure according to the needs of the prediction task to be solved, i.e. soft sensor to be built. Further, it was demonstrated that this can be done by using a default parameter setting without any manual parameter optimisation for the particular data sets. The soft sensors developed in this way were all considered cases able to provide useful predictions and in several cases even achieved outstanding performance.

The presented results demonstrate that the complex soft sensing algorithm is an important step towards the fulfilment of the transferability, i.e. the ability to deploy useful soft sensors with minimal effort of the soft sensors, which is one of the major goals of this work.

### 6.6.5 Minimal training data soft sensors

The experiments presented in this section aim at the analysis of the ability of the soft sensing algorithm to develop adaptive soft sensors starting with a minimal training data set. The collection of a large amount of training is required by most of the current soft sensors considered in Chapter 2. Some of the extreme cases require the collection of historical data over *several months* of operation of the processes.

To analyse the ability of the algorithm to deal with this constraint the soft sensors are developed using only $10\%$ of the available data as historical data. After the initial training, the soft sensors are deployed. For the on-line data, only $25\%$ of the target values are used for the adaptation. For all of the soft sensors, the default parameters from Table 6.1 are applied.

#### Industrial drier

The predictions of the industrial drier soft sensors developed using only a minimal amount of training data are shown in Figure 6.36. One can see that the on-line phase takes longer (compare e.g. with Figure 6.31) as it now covers $90\%$ of the data set. The soft sensor is able to deliver useful

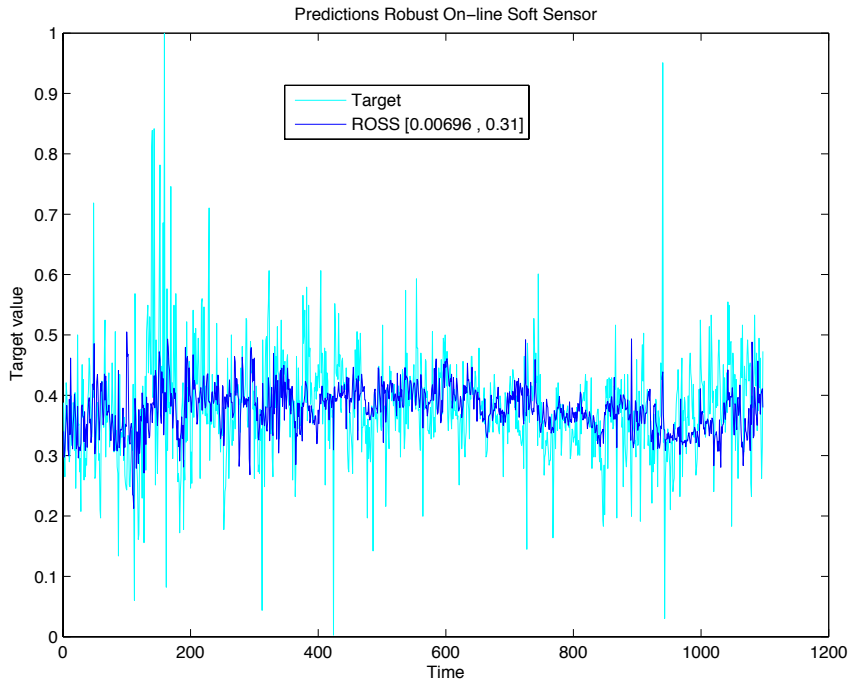



Figure 6.36: Industrial drier: Predictions of a soft sensor developed using minimal training data

predictions from the beginning of its operation (see Figure 6.37), the on-line area that is covered poorly is between samples 130 an 180, which can probably be attributed to the high noise level of the target variable in this range.

In order to analyse the influence of the limited training data, Table 6.8 shows the MSE and correlation coefficient of the full scale soft sensor from Section 6.6.4 next to the performance of this soft sensor measured over the same on-line data samples. The table shows that the minimal training data soft sensor's performance is equivalent to the full-scale ROSS using the full training data. This clearly demonstrates that the limited amount of training data is no obstacle for the soft sensor because it maintains a stable performance by means of its adaptation.

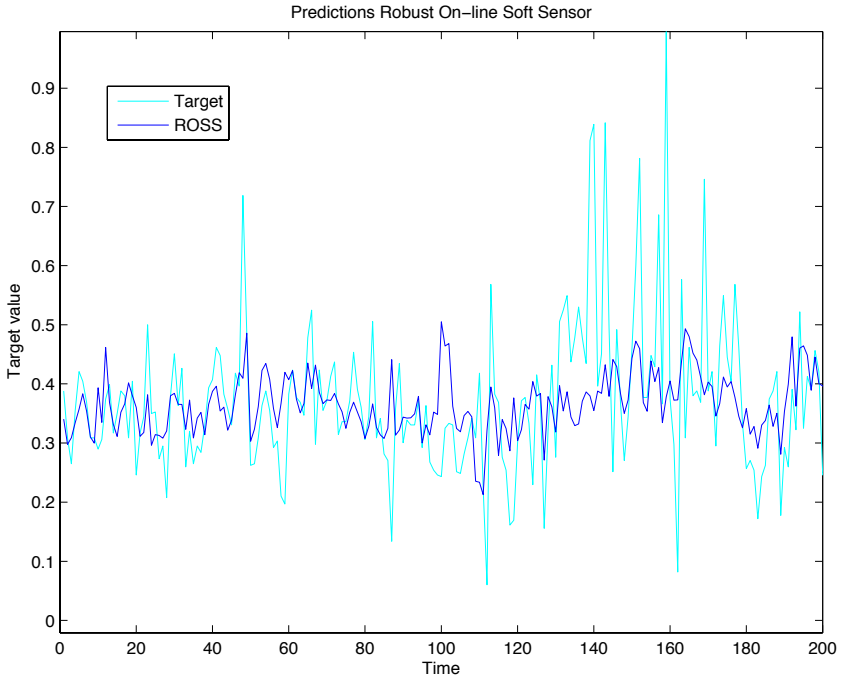



Figure 6.37: Industrial drier: Minimal training data soft sensor predictions (detail view of the first 200 samples)

| Soft sensor type | MSE | Corr. coef. |
|---|---|---|
| Full-scale ROSS | $5.35 * 10^{-3}$ | 0.26 |
| Minimal training data soft sensor | $5.20 * 10^{-3}$ | 0.28 |

Table 6.8: Industrial drier: Comparing the minimal training data soft sensor with the full-scale soft sensor from Section 6.6.4

### Thermal Oxidiser

The overall predictions of the thermal oxidiser soft sensor developed with minimal training data is shown in Figure 6.38. One can see that this soft sensor is also able to deal with the on-line data. However, unlike in the previous case, this soft sensor has problems at the beginning of the training phase as shown in Figure 6.39. Nevertheless, after the initial problems the model stabilises, which is also confirmed in Table 6.9, which shows its performance over the last 1437 samples, i.e the range equivalent to the on-line phase in previous thermal oxidiser experiments. The table reveals that both the full-scale soft sensor from Section 6.6.4 and the soft sensor trained with minimal training data achieve similar performance and thus demonstrates the ability of the algorithm to start with only minimal training data and to continue to learn during the learning phase.

| Soft sensor type | MSE | Corr. coef. |
|---|---|---|
| Full-scale ROSS | $9.91 * 10^{-4}$ | 0.64 |
| Minimal training data soft sensor | $8.77 * 10^{-4}$ | 0.69 |

Table 6.9: Thermal oxidiser: Comparing the minimal training data soft sensor with the full-scale soft sensor from Section 6.6.4

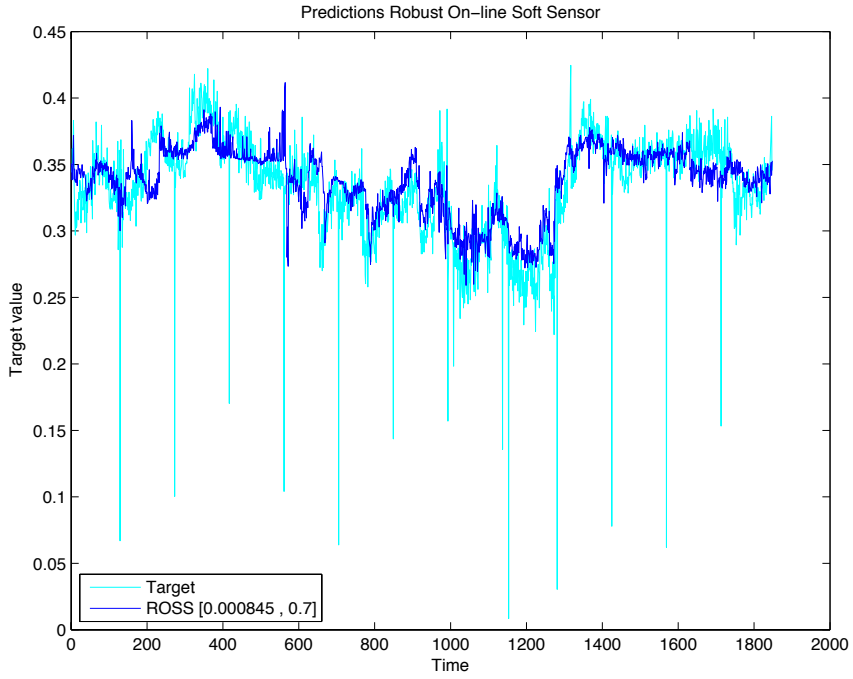



Figure 6.38: Thermal oxidiser: Predictions of a soft sensor developed using minimal training data

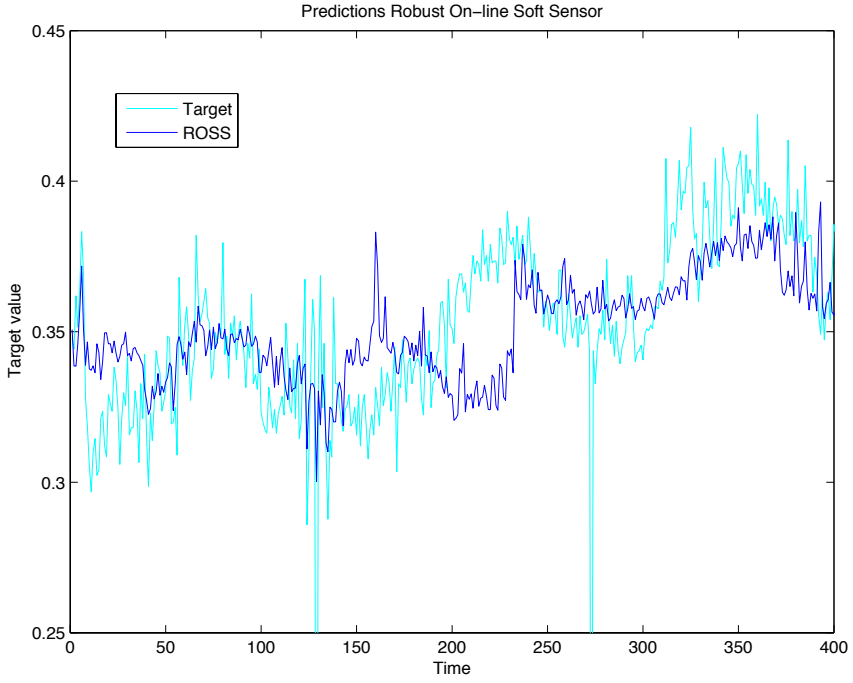



Figure 6.39: Thermal oxidiser: Minimal training data soft sensor predictions (detail view of the first 400 samples)

### Catalyst activation

The predictions of the catalyst activation soft sensor for the extended on-line phase are shown in Figure 6.40. The soft sensor seems to have a problem with the predictions of the first 200 samples,

which is confirmed in Figure 6.41, which focuses on this part of the on-line phase. The reason for this is probably the extremely low number of data points available for the training, which, in the case of this experiment, are only 65 samples.

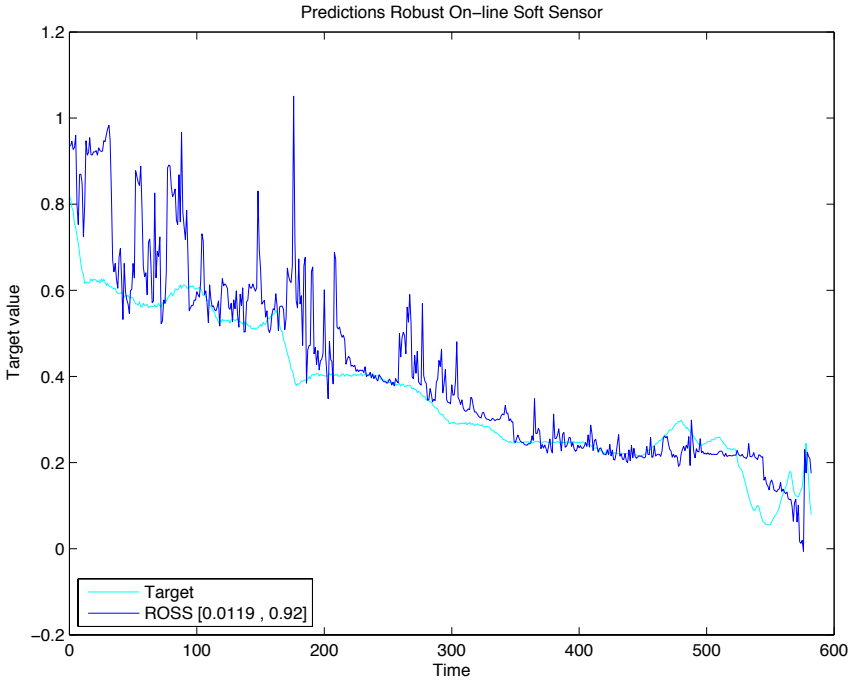



Figure 6.40: Catalyst activation: Predictions of a soft sensor developed using minimal training data

An interesting fact can be observed in Table 6.10 (as well as in Table 6.9 and Table 6.8), which compares the performance of the soft sensor from Section 6.6.4, which is trained with the full training set, with the soft sensor presented in this section. It turns out that the soft sensor trained with only $10\%$ of the samples achieves much better performance than the one trained with the full training data, i.e. $30\%$ of the data. The reason for this paradoxical observation is that the on-line adaptation is probably more effective (even despite the fact that only $25\%$ of the data are used for adaptation) than the intitial training process.

| Soft sensor type | MSE | Corr. coef. |
|---|---|---|
| Full-scale ROSS | $1.18 * 10^{-2}$ | 0.78 |
| Minimal training data soft sensor | $6.37 * 10^{-3}$ | 0.87 |

Table 6.10: Catalyst activation: Comparing the minimal training data soft sensor with the full-scale soft sensor from Section 6.6.4

### Experiment conclusion

The experiments with the minimal training data have shown that the limitation can have a negative effect on the prediction for the early data points in the on-line phase. This issue is caused by the small amount of the initial training data. However, all of the considered soft sensors managed to deal with poor prediction performance for the early data points by means of on-line adaptation

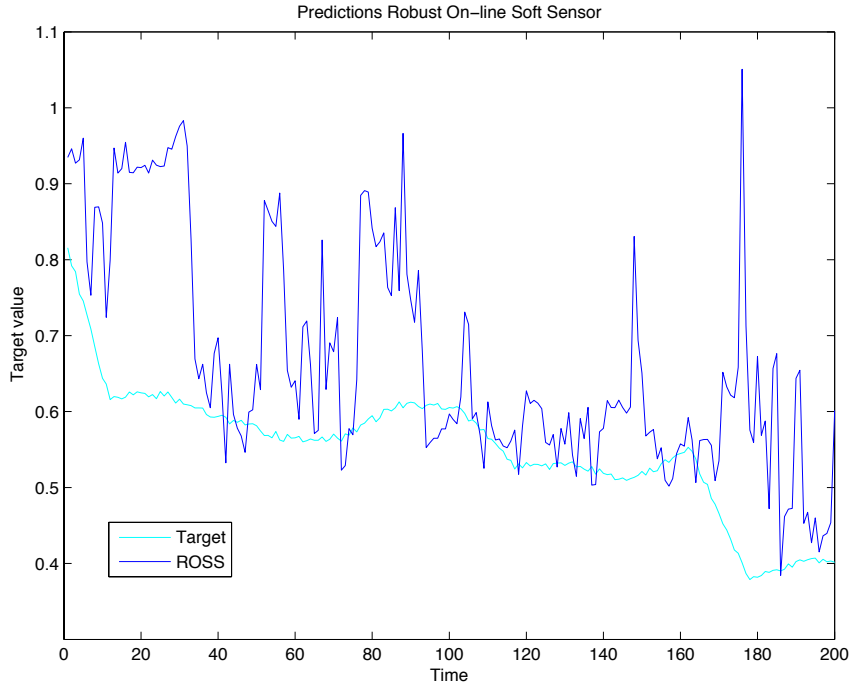



Figure 6.41: Catalyst activation: Minimal training data soft sensor predictions (detail view of the first 200 samples)

and over comparable ranges delivered similar or better performance compared to the soft sensor presented in Section 6.6.4, which use the full training data set.

## 6.7 Summary

The main contribution of this chapter is the demonstration of a practical instantiation of the abstract architecture presented in the previous chapter. The implementation, i.e. the complex soft sensing algorithm, shows that by following the structure of the architecture, a flexible, adaptive and robust algorithm for the development of soft sensors can be constructed. The algorithm's core is built by the adaptive local learning discussed in Chapter 4. The extended algorithm relies exclusively on established and proven machine learning principles such as cross-validation, boosting and ensemble methods. Although very complex, the algorithm is easily manageable and provides mechanisms that allow off-the-shelf deployment of soft sensors without any manual parameter or model selection.

Next, soft sensors for the three process industry data sets used in Chapter 4 were developed using the complex soft sensing algorithm. The experiments focus on aspects like the analysis of the influence of the number of local expert on the performance of the soft sensor. Another aspect that was analysed is the influence of the amount of feedback available for adaptation purposes. It turned out that for two out of the three data sets a small amount of feedback, i.e. target values, is already sufficient to maintain a stable performance level. In another set of experiments the ability of the algorithm to develop simple and transparent soft sensors was demonstrated. These soft sensors were a locally valid convex combination of a few linear models. Although the soft sensors delivered sub-optimal performance, the performance level was still acceptable and similar to the LWPR-based soft sensors. In the experiment, the ability of the algorithm to develop soft sensors

off-the-shelf, without any manual intervention from the user, was analysed. The conclusion of this experiment was that the algorithm is able to adapt the structure of the resulting soft sensor to the underlying data set, i.e. modelling task. The final set of experiments paid attention to the ability of the algorithm to develop adaptive soft sensors with a minimal amount of training data. Although two of the soft sensors had difficulties at the beginning of the on-line phase, they managed to improve their performance during the on-line phase by means of on-line adaptation.

# Chapter 7

# Conclusions

## 7.1 Project summary

The primary target of this work was to define a concept for the development of next-generation soft sensors. The purpose of the concept is to help to overcome the major sources of frustration with practical implementations of soft sensors in the process industry.

In order to be able to address this problem, it was first necessary to review the current state-of-the-art soft sensor development and application. In this work this task was approached from two different directions. On one hand a comprehensive review of a large number of academic articles dealing with practical implementations of soft sensors was done. On the other hand, valuable practical information was obtained from discussions and interviews with experienced soft sensor developers from Evonik Degussa GmbH. Both of these aspects are reflected in Chapter 2. Although there are many differences in these two view-points, one point where both agree is that there is a large amount of process a-priori knowledge necessary to be able to develop useful soft sensors. This knowledge has to be applied to data pre-processing, e.g. variable selection, or model type and parameter selection, which makes this step time consuming and costly. A particularly critical aspect that is neglected in the vast majority of publications is the soft sensor maintenance issue. In most publications the developed soft sensor is applied only to data that covers only a few months of the process operation, which is considered as sufficient to demonstrate that the soft sensor performs well. However, in practical scenarios the soft sensors are required to operate over much longer periods of time, which is often prevented by effects like process dynamics or changing data quality. In fact, the soft sensor maintenance can be much more expensive than its development because the model has to be re-trained or re-developed periodically. This is a very limiting factor requiring a lot of attention from the soft sensor operator and prohibiting a wider spread of soft sensors in the process industry. After this review, it was clear that the developed concept has to facilitate not only quick and efficient development of soft sensors but has also to pay at least the same attention to the automation of maintenance of the soft sensors in the form of their self-adaptation.

The next step was the identification of useful machine learning concepts and techniques, which, theoretically, could be useful for achieving the project goals. What became obvious at this stage was that the work had to be embedded in the concept of *algorithm independent learning*. This fact was supported by the previously discussed review, which has shown that there are many different methods, not only for modelling but also for data pre-processing, applied to soft sensor development. Each of these methods has its strengths and weaknesses and there is no *golden* method to which this work could be restricted. One of the first concepts that turned out to be

potentially beneficial was *local learning*. By applying this technique it is possible to train models, called local experts, for different operational states of the processes. An inevitable challenge that emerges from this concept is the switching or combining of the different local experts. In general, this goal can be approached using *ensemble methods*, which is the next class of techniques and was reviewed in Chapter 3. The effectiveness of ensemble methods is backed-up by analytic and empirical evidence, which was another motivation to consider this concept. As it started to be apparent that the developed concept for soft sensor development will be rather complex, another technique that would handle its high-level aspects had to be found. In this context, *meta-learning* turned out to be very useful. Meta-learning techniques can be used, for example, for extraction and transfer of high level knowledge. All of the methods mentioned so far focus mainly on non-adaptive, off-line predictive modelling. To close this gap to the on-line adaptation, *concept drift detection and handling* was reviewed. This research topic provides the theory as well as many practical techniques for dealing with adaptation aspects of predictive models. Another benefit is that it is largely independent of the underlying predictive methods and can easily be combined with the techniques like local learning and ensemble methods, i.e. it fits well into the algorithm independent learning framework.

The local learning and ensemble methods together together with a specific concept drift handling method were applied to develop an early-stage algorithm presented in Chapter 4. The goal of the algorithm was to prove that these approaches are useful for soft sensor development and dealing with the issues identified earlier in this work. The local experts applied in the algorithm were simple models consisting of local PCA pre-processing and linear regression techniques. They were trained on partitions of data that were segmented using a novel approach. The experimental evaluation was done using data sets from three different processes from the industrial partner of the project. The applied methods included apart from the novel algorithm (Local learning-based Adaptive Soft Sensing Algorithm- LASSA) also LWPR, which is another local learning-based technique, as well as traditional soft sensors based on ANN. The experiments have shown difficulties with applying ANN to the real-life data sets. It was proven that it is not only difficult to select correct parameters, but also that after the selection, the performance of the optimally parametetrised ANNs can still vary significantly. This is mainly due to the random initialisation and local minima problems of the ANNs. Another important result from the experiments was the good manageability and strong performance of the local learning-based methods. Another particularly important feature of these algorithms was their modularity and the possibility to apply adaptation mechanisms to the algorithm. The LASSA adaptation mechanism was based on the modification of the combination weights of the local experts. This was shown to be effective for two of the three data sets. However, for the third one, namely the catalyst activation, this approach was not able to adapt the soft sensor adequately. The reason, why this method failed was because it did not support deployment of new models (local experts) during the on-line phase and it merely adapted the contributions of the existing local experts to the final prediction. The final conclusion of the experiments was that the local learning approach combined with ensemble learning is very useful for the development of adaptive soft sensors but, in order to offer a truly adaptive soft sensor, more than this simple adaptation method had to be provided.

All of the findings and experiences with soft sensor development were projected into the architecture for the development of soft sensing algorithms presented in Chapter 5. The machine learning techniques discussed in Chapter 3 were arranged into a three-level hierarchical structure where each of the levels represents a different type of information processing. The bottom level, where the actual prediction making units operate, represents the lowest complexity but largest diversity level. At the next level, the predictions of the models, e.g. local experts, are combined and

thus more complex predictors are built. Although the diversity at this level is much lower than at the bottom level, it is still present in the form of multiple combination schemes. At the top level, the complexity level reaches its maximum and there is no more diversity present. From this level, the operations at the lower levels are managed. The adaptation loops of the architecture follow its three-level structure and the implemented mechanisms can range from the adaptation of the models at the bottom level through the adaptation of the combinations at the intermediate level to the high-level adaptation where new combinations or local experts are deployed. The next important aspect of the architecture is the role of expert knowledge. In practical scenarios there is often some kind of expert knowledge about the underlying process available. In such cases it is crucial to provide a formal mechanism for the incorporation of this knowledge into the soft sensors. At the present stage the interface for the expert knowledge incorporation is only defined theoretically.

The final step of this work was the presentation of a complex soft sensing algorithm that was developed by following the architecture. The core of this algorithm is the local learning algorithm presented earlier in this work. The extensions presented in Chapter 6 aimed at increased robustness and more effective adaptiveness of the resulting soft sensors. This is achieved by: (i) extending the available pre-processing and modelling techniques; (ii) building several local experts per receptive field; (iii) providing advanced selection and pruning of the local experts; (iv) using advanced data management based on two-fold cross-validation; and (v) providing several adaptation mechanisms. In particular, there is special attention paid to the adaptation mechanisms. There is at least one mechanism implemented at each of the hierarchy levels providing the algorithm with the ability to follow the changes of the on-line data as well as the ability to adapt the algorithm to the current task. The subsequent experiments focused on the following aspect of the complex soft sensing algorithm: (i) the role of the local expert population setting and its influence on the performance of the soft sensors; (ii) the behaviour of the algorithm with varying percentage of target data available for adaptation; and (iii) the fulfilment of the project goals. Consequently, it was shown that the algorithm's adaptation mechanisms were successful in updating the models. Another experiment demonstrated that despite the large number of input parameters the algorithm achieved good performance even if applied off-the-shelf using its default parameter settings. The next set of experiments demonstrated that, by exploiting the flexibility of the algorithm, it can be used for the development of low complexity soft sensors. Although the performance of the low complexity soft sensors was below the performance of the complex models, it was still acceptable and in fact similar to the performance of the benchmark method (LWPR-based soft sensors). An important aspect of the algorithm that was also experimentally evaluated was the its ability to start with a minimal training data set. This is particularly important aspect for the practical application of the algorithm because, if successful, it would help to avoid the costly training data collection required by current techniques. The results were very encouraging since the models either performed very well from the beginning or, in cases where the training data set was too small to train a useful model, managed to improve their performance during the on-line phase by means of the implemented adaptation mechanisms.

## 7.2 Achievement of the set goals

This section discusses the way in which the goals of this project set in Section 1.3 were approached as well as evidence of their fulfilment.

### 7.2.1 Simplified soft sensor development

The presented architecture provides a set of mechanisms that support this goal. A significant role is in this case played by the pools of pre-processing and modelling methods (PPMP and CLMP respectively) together with the Path Control, Path Combination Control and Meta-Level Learning, which support the selection of suitable methods. The goal of the interactions between these mechanisms is to train, select and maintain useful models without requiring any input from the model developer.

In terms of the complex soft sensing algorithm presented in Chapter 6, there are several mechanisms that manage a large part of the soft sensor development. These include the possibility to define parameter ranges and the provision of optimisation mechanisms that find their optimal (within the range) values (see Section 6.4). The algorithm can also be trained using a default parameter set-up that does not require any manual intervention from the user at all.

The above facts were evaluated in several experiments, where the soft sensors developed using the soft sensing algorithm were built with minimal manual user intervention. In particular, the transferability experiments in Section 6.6.4, have shown that the algorithm is able to fit the structure of the soft sensor to the underlying data. A similar effect was shown in the case of the low complexity experiments, where three different soft sensors were also developed without any manual parameter or model selection done by the user.

### 7.2.2 Prolonging the soft sensor life-time

The arrangement of different adaptation loops that fit into the three-level hierarchy of the architecture is one of the main contributions of this work. The adaptation aspects of the architecture are discussed in detail in Section 5.4. The aim of the adaptation mechanisms, which can be implemented in the loops, is to allow the soft sensors to deal with the process dynamics and the changing environment represented by the process industry data and thus to prolong their life-time.

The soft sensing algorithm from Chapter 6 implements at least one adaptation mechanism for each of the architecture's adaptation loops, as shown in Section 6.5. The mechanisms range from a low-level adaptation provided by the LWPR technique to complex deployment of new receptive fields and local experts involving the transfer of meta knowledge.

In terms of the application of the algorithm, the adaptation aspects were targeted in the experiments in Section 6.6.2. The results have shown that the soft sensors clearly benefit if there is feedback information provided for their adaptation, which in turn demonstrates the effectiveness of the adaptation mechanisms. Another experiment where the adaptation capability turned out to have a positive effect on the soft sensors was in the 'minimal training data' experiments in Section 6.6.5, where the models improved their initially low performance by the means of on-line adaptation.

### 7.2.3 Flexibility of the soft sensing algorithm

The proposed architecture is undoubtedly very complex, which can potentially lead to some issues for the resulting soft sensors. However, it also has the advantage of providing flexibility. As the architecture defines only a framework for the development of soft sensing algorithms, the mechanisms implemented within its particular modules are entirely up to the designer of the soft sensing algorithm.

The instantiation of the architecture in the form of the soft sensing algorithm presented in Chapter 6 aims at the demonstration of the type of mechanisms that can be implemented at differ-

ent positions of the architecture. Like the architecture, the algorithm is also rather complex with a large number of input parameters.

The flexibility was exploited in the experiments in Section 6.6.3, which demonstrated the ability of the algorithm to build simple soft sensors in a complex environment. This was achieved by modifying the parameters related to the complexity of the resulting soft sensors, for example $n^{LE,target}$ or $n^{LEC}$, and by allowing only the multiple linear regression modelling technique. The flexibility of the algorithm allowed the development of well performing simple soft sensors, as the algorithm managed to fit the structure of the resulting models to the underlying data.

### 7.2.4 Incorporation of expert knowledge

This goal has been achieved at the level of the architecture, which dedicates the Expert Knowledge module to this aspect. It also defines the connections, i.e. interfaces, to the other modules as shown in Figure 5.5. Using these interfaces, the expert knowledge can be used to intervene at any given part of the architecture.

In the case of the experiments presented in this work, the expert knowledge was implemented into the models in an ad-hoc way by manually pre-processing the data as described in Appendices B.1-B.3.

The practical implementation of the Expert Knowledge module providing user interaction possibilities with the soft sensing algorithm will be the subject of further research resulting from this work.

## 7.3 Main findings and contributions

The main contributions of this work are:

- Comprehensive review of data-driven soft sensors:

  This work provides a rigorous review of the current state of data-driven soft sensor development and maintenance. The review lists a large number of practically applied soft sensors. It presents not only current trends in soft sensing but also the most significant issues that prevail in this research discipline. It was found that many issues can be traced back to the data upon which the soft sensors are built. The current practice for dealing with these issues is to collect and apply as much process knowledge as possible. However, this practice is very costly, which in turn obstructs larger proliferation of soft sensors in the process industry. The review was published in a separate article in [96].

- Review of machine learning concepts used in this work:

  This review focuses on the machine learning concepts that support achieving the goals of this work. In particular, the reviewed concepts and techniques are: (i) ensemble methods; (ii) local learning; (iii) meta-learning; (iv) concept drift detection and handling; and (v) predictive methods for data-driven soft sensing. The interesting result is that the different concepts are more interlinked than it may initially appear. For example, ensemble methods can be found as a particular solution in local learning, meta-learning and concept drift detection and handling.

- Random Topology Gating Artificial Neural Network (RTGANN) and Learnt Topology Gating Artificial Neural Network (LTGANN):

These two meta-learning algorithms were developed during the early stages of this project and presented in [92] and [93], respectively. Both of the algorithms experiment with the application of ensembles of neural networks to process industry data. Within the algorithm, the combinations of the outputs are performed by gating networks that weight the predictions of the base networks according to their estimated prediction accuracy. In the case of LTGANN, the algorithm additionally learns the optimal topology of the networks added to the ensemble. The results have shown that in both cases, the prediction performance was improving with increasing number of ensemble members until it converged at a stable level that outperformed the benchmark models.

- Novel Local learning-based Adaptive Soft Sensing Algorithm (LASSA):

  This algorithm is conceptually related to Locally Weighted Projection Regression (LWPR) [177]. The major distinguishing aspects of LASSA are: (i) the data partitioning, i.e. receptive field building, algorithm; (ii) application of PCA- rather than PLS-based projection to low dimensional spaces; (iii) application of Parzen windows rather than Gaussian function for the receptive field descriptors. A particular benefit of the algorithm is its modular and open structure, which was exploited during the later stage of the project.

- Empirical evidence of the effectiveness of local learning techniques for soft sensor development and maintenance:

  As part of this work, local learning based algorithms were applied to soft sensing. By applying the algorithms to three real-life predictive modelling tasks from the process industry, their effectiveness was demonstrated in terms of the application of the these methods to three industrial data sets. The algorithms achieved strong prediction performance, were easily manageable in terms of their parameter selection and incorporated adaptation mechanisms that allowed to prolong the life-time of the resulting soft sensors. This evidence can also be found in two published conference papers [91, 94].

- Architecture for the development of robust and adaptive soft sensors:

  The architecture defines a framework for the development of robust and adaptive algorithms. In particular it was developed with focus on the construction of adaptive soft sensing algorithms. The architecture unifies the machine learning concepts like local learning, ensemble methods, meta-learning, etc. into a complex structure. The robustness of the resulting algorithms is achieved by providing a highly dynamic environment where predictive models and their combinations can be launched, adapted and deleted dependent on their performance, diversity, etc. The adaptive behaviour is supported by the definition of multiple adaptation loops that can accommodate different adaptation mechanisms. Another unique characteristic of the architecture is the definition of interaction channels, which can be used by the model developer/operator to implement expert knowledge into the models as well as manually intervene in the operation of the models.

- Robust and adaptive On-line Soft Sensing algorithm (ROSS):

  This is a complex adaptive algorithm that was developed in the framework defined by the architecture. The algorithms consists of advanced data management attempting an effective exploitation of the data during the training as well as the on-line phase. It also features autonomous selection mechanisms for the data pre-processing and predictive modelling techniques, which are selected from pools of available methods. The algorithm also accommodates five different adaptation mechanisms acting at different levels of the architecture's hierarchy and effectively extending the life-time of the models. The algorithm is evaluated by its application to the three industrial data sets. The architecture together with the ROSS were published in [95].

## 7.4 Further research topics

One of the most significant characteristics of the proposed architecture is its modular structure. The possibilities of the architecture are so large that many aspects could not be targeted due to the time constraints of this project. Many of the untouched aspects are planned to be targeted during further research activities, which are at the moment supported by both the industrial partner as well as the academic side of the project.

The following list provides a brief discussion of the most important research topics around the architecture:

- Pre-processing and predictive techniques: This research should deal with the identification and development of new pre-processing and predictive techniques supporting soft sensing or any other application of the architecture.

- Further adaptation mechanisms: The conceptualisation of the different levels of adaptation mechanisms is one of the strengths of the proposed architecture. Although the implementation in this work shows examples of adaptation strategies at different levels of the architecture, there can be many more strategies implemented.

- Data management strategies: Effective data management within the architecture is important for the maximal exploitation of the data, which is often available in very limited amounts. The proposed complex soft sensing algorithm shows a possible way for the handling of the available data mainly based on two-fold cross-validation and boosting-like sampling of the data. The optimisation of data management during the off-line and on-line phases and the development and implementation of advanced data handling techniques in the context of the architecture will be subject of further studies.

- Mechanisms for management of the local experts and their combinations: Another important aspect of the architecture is the management of the population of models, e.g. local experts, and their combinations. In the presented implementation, these mechanisms are based on the competitive, diversity and collaborative selections. Although these mechanisms have been shown to be successful in the context of the studied soft sensors, more sophisticated mechanisms for balancing the diversity and performance of the local experts can be accommodated.

- Predictions post-processing: In some applications the post-processing of the predictions appears to be as important for the model performance as the data pre-processing. Within

the architecture, the post processing can be implemented as part of the Global Evaluation module.

- Model transparency: One of the most important aspects of industrial applications of predictive modelling is the transparency of the models and of the decision making inside the models. In order to maximise the acceptance of soft sensors among the industrial practitioners, it is necessary to equip the models with the ability to demonstrate how it derives the predictions and other important decisions. This aspect is especially required for the troubleshooting of the models in cases when the model fails. In terms of the architecture, the transparency can be particularly challenging due to its complexity and therefore the research should focus on a meaningful, possibly high-level, explanation of the operation of the models. Offering a solution for this problem could help to achieve higher acceptance of soft sensors in the process industry.

- Optimisation and management of computational resources for time critical applications: In the case of soft sensors application, the time needed for prediction and adaptation of the models is not critical because the time intervals between the provided data samples are long enough to fulfil these tasks. For this reason, the management of computational resources related to prediction and adaptation time has not been dealt with. However, in other practical scenarios where these aspects may play a critical role, there will be mechanisms for real-time operation necessary.

- Treatment of the architecture as a complex system: The architecture can be treated as a multi-level complex system that allows the application of results from complexity science research. Complexity science can offer mechanisms for dealing with cross-level aspects of the architecture. This involves, for example, the study of the influence of changes made at the lower levels of the platform on the performance level of the whole model. Another topic that can be researched is the inter- and intra-level interaction of the components and the influence of the interaction on the dynamic behaviour of the platform. Another topic that can be dealt with is the study of the dynamics of adaptation (from very fast to very slow) of the individual methods and their influence on the combined predictors adaptation and stability aspects resulting from the adaptation.

- Dealing with the incorporation of expert knowledge: For practical applications, expert knowledge incorporation may be necessary.

- Application to batch-type processes: Batch processing plants briefly discussed in Section 2.2.2 have some specific features that differentiate them from the continuous processes handled. The main difference is the definite, often rather short, operation of the process and the batch to batch variations. In order to be able to deal with these challenges, modification of the way the data is treated is required.

- Applications to other industrial problems: The demand for predictive modelling is larger than ever. There are many industries where large amounts of data are being collected. At the same time there are a lack of approaches that can support full exploitation of the latent potential of the data.

- Professional implementation: In order to make the concept and the resulting soft sensing algorithm applicable in the industry, it requires a professional implementation. At the moment, the implementation is an experimental prototype implemented in Matlab. The proto-

type served well as proof of concept presented in Chapter 6, but for a real-life application it requires re-design and re-development in an industrially relevant programming environment. Apart from the coding, there is also a need for research of suitable software engineering concepts that are suitable for the implementation of the architecture. This includes the definitions of interfaces between the modules, i.e. for the data, control, evaluation and pooling connections, as well as the definition of the objects like computational path, path combination, evaluation object, etc. In particular, the chosen approach should support the development of the open and modular structure of the architecture.

# Appendix A

# Lists of soft sensor applications

## A.1 On-line prediction applications

Linear regression models are the most straightforward way of modelling the target values. In this case, the modelled variable is a linear combination of the input variables.

A soft sensor for the modelling of the particle size in a grinding plant was published in [22]. The developed soft sensor is an ARMAX-type stepwise regression model. The input for the model are systematically selected based on the correlation between the analysed input feature and the output, including delayed versions of the input variables. The authors present a set of models using different types of input including combined inputs based on the process (phenomenological) knowledge about the process. The best performance is achieved by a model combining historical data and physically significant combinations of the input variables, i.e. a grey-box model.

Locally Weighted Regression (LWR) together with non-linearity handling pre-processing are applied in [134]. As the process data are non-linear, the authors propose to use models with a limited field of influence (local models). The advantage of these kind of models is that one can use less complex linear models to deal with the problem. The performance of the proposed soft sensor is compared to other common modelling approaches like ANN in terms of two industrial data sets (toluene composition in a splitter column and diesel temperature estimation in a crude oil column). The results show that the LWR-based method provides comparable or better results when compared to the other modelling techniques.

As an output of this work a local learning-based soft sensor was published in [91]. This soft sensor is based on a combination of locally valid models. These local models are combinations of ten Multiple Linear Regression (MLR) models. The receptive fields are modelled using the Parzen window technique. Based on an application of the soft sensor to an industrial thermal oxidiser process, the model shows much better performance than a traditional MLP-based soft sensor. Furthermore, the presented approach provides several possibilities for adaptation of the soft sensor, which leads to further performance improvement.

Another typical modelling approach used for these problems is the application of Multi-Layer Perceptron (MLP), which is one of the most popular Artificial Neural Network (ANN) models used for function approximation.

Thorough analysis of the application of MLPs for soft sensor building has been presented in [144]. This work discusses a lot of practical issues of the application of neural networks for soft sensor modelling. A particular focus is put on the necessary pre-processing steps like the handling of missing values and outliers. Focusing on the identified issues, there is also a modification of the error measure of the back-propagation algorithm (i.e using Manhattan distance instead of mean

squared error) proposed. Furthermore, the MLP-based soft sensor is compared to an NNPLS model. Based on a case study dealing with a batch refinery process, it is shown that the NNPLS outperforms the MLP due to better generalisation performance and more effective dealing with data co-linearity.

In [90], an MLP is compared to model-driven approaches based on First Principle Model (FPM), adaptive observer technique and extended Kalman Filter (eKF) models, which are common approaches to model-driven soft sensor building. The disadvantages of FPM and eKF are the complexity of the development and amount of process knowledge that has to be available for the model development. On the other hand, the applicability of the MLP for solving on-line estimation of fermentation batch processes is limited due to the changing dynamics of the particular batch runs. The authors therefore suggest a hybrid solution where the process dynamics is described by a model-driven model and the MLP black-box approach is used to model only parts of the model, like the growth rate of bioprocesses.

A grey-box soft sensor that delivers necessary control information for self-tuning adaptive controller of a fermentation process was presented in [127]. The soft sensor is an MLP, which is trained using simulated data based on a phenomenological model of an ethanol production plant. After training, the model is validated using industrial process data. The soft sensor is successfully implemented into the control loop of the process controller.

An extensive discussion of application aspects of MLP to steel industry data modelling was published in [147]. They provide a detailed procedure including data pre-processing, model selection, etc., for the application of MLP to the modelling of metal quality in a blast furnace. Also presented is an expert system for the control of silica content. In a real-life application, the installation of the soft sensor and the expert system leads to significant improvement of the steel production.

An application of MLP for sugar quality estimation was published in [40]. The problem approached in this work is the modelling of the massecuite electrical conductivity, which is an important value for the control loop controlling the sugar production process. The eight input features of the model were selected manually using process knowledge about the process. The results achieved by the MLP were good enough to take the soft sensor into real-life operation.

A complex soft sensor based on MLP was developed and published in [56]. The soft sensor models butane and stabilised gasoline concentrations of a distillation column. The model is a cascaded 3-level neural network. Apart from the input variables, which are measurements within the column, the model uses delayed versions of the input variables. The model gives satisfactory results for the on-line prediction of the concentrations.

The performance of two ANN variants, namely the MLP and the Radial Basis Function Network (RBFN), are compared to a Support Vector Regression (SVR) model in [39]. The data sets for the comparison are two simulated batch bioprocesses. It is clearly shown that the performance of the SVR soft sensor is superior in comparison to the other two methods. The authors also provide a theoretical explanation of the performance benefits. The ability to locate global minima of the presented problems and the interpretability of the learnt knowledge in terms of the training data (support vectors) are stated as advantages of the proposed SVR soft sensor.

Another performance comparison between MLP and RBFN was published in [85]. In this work, these two model types are also compared to a grey-box model based on a first principle model and either an MLP or an RBFN. The performance was tested in terms of a biomass concentration prediction in a biochemical batch process. The hybrid model is described as the best performing one. However, the performance gain comes at the cost of process knowledge that has to be input into the model.

In [185], an RBFN-based soft sensor for the modelling of a membrane separation process was developed. The Multiple Input Multiple Output (MIMO) soft sensor predicts some critical process performance values (like gas concentrations). The aim of the soft sensor is to deliver additional on-line information for process control.

An ensemble approach for soft sensor development based on MLPs was published in [93] (also a publication resulting from this work). In this work the problem of optimal network complexity selection was approached in the context of ensemble methods. The optimal MLP topology was established by training several models with different complexities and assessing their relative performance. In such a way, performance distributions across the different parameter values were calculated. The final ensemble is built by weighting the contributions of ensemble members by their estimated generalisation performance. This soft sensor was applied to an industrial drier process.

An application of Recurrent Neural Network (RNN) to the modelling of the degree-of-cure, which is an important quality indicator in an epoxy/graphite fibre composites production process, was published in [166]. The soft sensor is a grey-box model, making partial use of process information about the process. The soft sensor was parametrised, trained and evaluated using simulated process data and, after some minor tuning, it was tested using real process data and target values obtained from off-line laboratory measurements. The authors were satisfied with the performance of the soft sensor and deployed it in the real-life process environment.

An RNN was also applied to the prediction of biomass concentration in [28]. RNN was applied in this work due to its theoretical ability to capture dynamic effects underlying the data. Although the RNN model performance is not compared to any other model type, the authors conclude that recurrent artificial neural networks are capable of achieving a satisfactory prediction performance.

Another RNN application to the prediction of the melt-flow-length for filling of molds in the injection molding process was presented in [29]. The authors decided to use the recursive version of ANN because of its capability to store temporal patterns, which is of advantage in the modelled process. The developed soft sensor provides accurate results of the melt-flow-length prediction.

Focused on soft sensing in a dynamic environment, the authors discuss the application of a multi-step predictor and decide to use an RNN for its implementation in [202]. They are using an Inner Recurrent Neural Network, where only the hidden layer has recursive connections. The usefulness of the algorithm is demonstrated based on three dynamic simulated processes.

The authors of [53] propose a grey-box technique for the implementation of process knowledge in a data-driven model. They focus on ANN, which provides the possibility to deploy nodes (neurons) that represent the process knowledge, e.g. single differential equations, etc. The nodes are abstract signal processing units transforming the input information to their output using arbitrary, but differentiable, equations. The authors apply the proposed ANN to the estimation of diacetyl in a biochemical process.

Another method commonly applied to soft sensing is the PCA/PLS-based regression.

A self-validating soft sensor is presented in [146]. The input data is validated using a PCA-based approach for fault detection published in [50]. In the case of a detected failure, the sensor can be reconstructed using the correlation structure of the affected input measurement to the other input space variables. After this pre-processing step, which on one hand removes the co-linearity of the input data and on the other hand provides the ability for the reconstruction of sensor faults, a soft sensor using traditional modelling techniques is built. This soft sensor is successfully evaluated on a real-life problem dealing with air emission monitoring process data.

Dayal and MacGregor proposed a novel recursive version of the least squares algorithm based on the Exponentially Weighted PLS (EWPLS) [37]. The authors use an adaptive approach for the

time window length calculation. Within the time window, the samples are exponentially weighted dependent on their age. The model is successfully applied to two processes: a simulated continuous stirred tank reactor and an industrial flotation circuit.

Another recursive version of the PLS algorithm is devised in [145]. In this work the recursive PLS algorithm is extended to a version that works block-wise and is thus suitable for adaptive modelling. The algorithm is combined with the two common techniques for adaptive modelling, namely with the moving window and the forgetting factor approaches. The performance of the proposed algorithms is demonstrated by applying it to octane number modelling in a refinery process.

Application aspects of the PCA and PLS to the modelling of batch processes are dealt with in [204]. In the cited work, there is a set of PLS regression models using different regressors developed and evaluated. The data set used for the evaluation is a simulated distillation column. The PCA algorithm is used for the identification and discarding of erroneous process states. Due to the non-linearity of the process, the best prediction results are achieved using the multi-way PLS.

In [118], in addition to a systematic procedure for PCA-based soft sensor development, two case studies applying the proposed method to process industry problems, namely a free lime prediction and $NO_x$ prediction in a cement kiln, are presented. Within the proposed development procedure, firstly missing values are handled using an heuristic approach. This is followed by outlier detection using an univariate Hampel identifier and multivariate robust statistics, like the Q-Statistics and the Hotelling's $T^2$. After the data pre-processing, a PLS-based regression model performing a one-step-ahead prediction is derived.

In accordance to increasing popularity of SVMs in the machine learning community, there are also some recent applications of this technique to soft sensing.

A soft sensor based on SVR, or more accurately on Least Squares Support Vector Machines (LSSVM) is presented in [200]. The authors define an iterative procedure which, apart from involving the LS-SVM model, uses the Bayesian evidence framework for the optimal selection of the LSSVM model parameters. The model is successfully applied to the estimation of the freezing point of light diesel oil in a Fluid Catalytic Cracking (FCC) unit.

In [54], there is also an LS-SVM model applied to a process industry problem. The LS-SVM is chosen due to evidence for better generalisation properties when compared to an RBFN-based soft sensor. Indeed, the LSSVM outperforms the RBFN in the case study dealing with the prediction of gasoline absorption rate in a Fluid Catalytic Cracking unit. The LSSVM model is also described as being less dependent on the size of the training data set, providing stronger learning ability.

Another very popular and successful family of approaches applied to soft sensing are neuro-fuzzy models combining the advantages of ANNs, most commonly the multi-layer perceptrons, and Fuzzy Inference Systems (FIS).

A Neuro-Fuzzy System (NFS) model was developed and published by Wang and Rong in [190]. The presented NFS is trained using a two-step approach consisting of a clustering and a back-propagation algorithm. One of its advantages is that the connectionist structure is determined automatically. The proposed approach is applied to the modelling of a distillation column, more specifically, to the propylene purity modelling at the output of the column.

An example of this type of soft sensor is an ANFIS-based soft sensor applied to rubber viscosity prediction in [129]. Because there is no automated way to measure rubber viscosity, which is an important quality indicator, a soft sensor is necessary to deliver the data. In the publication, it is claimed that the accuracy of the soft sensor meets the requirements for its implementation in the process control loop.

Another ANFIS-based soft sensor was presented in [192]. In this work the data is pre-processed using PCA transformation, which on one hand helps to deal with the co-linearity of the data and on the other hand limits the size of the input space of the ANFIS model, which in turn reduces the complexity of the model significantly. The presented methodology is applied to the prediction of a polymeric-coated substrate anchorage, which is an important quality measure of the process product.

A neuro-fuzzy soft sensor based on rough set theory and optimized by a genetic algorithm is discussed in [121]. The rough set theory is used to obtain a reduced set of rules, which are then implemented in the form of an MLP. The genetic algorithm is used to get an optimal discretisation of the input variables. The performance of the algorithm is demonstrated in a refinery case study, namely on the prediction of freezing point of the light diesel fuel in a Fluid Catalytic Cracking unit.

Neuro-fuzzy FasArt and FasBack were applied in [4] for the modelling and control of a penicillin production batch process. A soft sensor for the prediction of the biomass, viscosity and penicillin production delivers the necessary information for the control mechanisms of the FasBack adaptive controller. The holistic control model is trained and evaluated using simulated process data. The trained model is then able to deliver satisfactory results for the real process control.

In [122] an extended Takagi-Sugeno (exTS) model has been applied to the prediction of the quality of crude oil distillation in a refinery process. The advantages of applying an evolving neuro-fuzzy model to this problem is reported to be the ability of the model to deal with non-linear problems and dealing with a large number of features. The presented model has the ability to evolve its rule base together with the dynamics of the process, which is an advantage of evolving neuro-fuzzy methods, distinguishing the NFS from other models.

Apart from the combination of ANN and Fuzzy Inference System (FIS), there is a large number of other hybrid models that are a combination of two or more computational learning techniques.

Qin's work published in [144] has already been mentioned. One of the contributions of this work is the definition of the Neural Network Partial Least Squares (NNPLS) algorithm, which is a hybrid system combining the PLS algorithm with an MLP. This algorithm makes use of the capabilities of the MLP to map the input variables non-linearly onto the latent variables of the PLS. The discussed hybrid algorithm is also applied to a refinery process.

Another application of NNPLS to soft sensing was presented in [43], where the NNPLS and the non-linear Principal Component Analysis (NLPCA) algorithms were applied to the prediction of emissions of $NO_x$ gas in exhaust streams. In this case, the input data is pre-processed by mapping it on principal components space using the NLPCA algorithm. After the pre-processing, the actual model, an NNPLS technique, predicts the target values. The application shows that the model outperforms a linear model and also demonstrates an immunity with regards to missing values.

A hybrid system consisting of Particle Swarm Optimisation, which is used for the training of an MLP, was presented in [116]. In this work the PSO algorithm is combined with the Alopex algorithm (see [170]) to avoid local minima to which the PSO is prone. The proposed algorithm is applied to an ethylene distillation column data set.

Another hybrid approach to soft sensor modelling has been developed by Kordon et al. [107, 108, 109, 110]. In this case, the hybridisation is done on a lower level. The involved methods perform pre-processing of the data for the succeeding modelling steps. The methodology for the inferential sensor building consists of three different steps. The first step is the analysis of the data by an analytical neural network [108]. The aim of this step is to perform feature selection on the

input data and to deal with time delays between the selected features. In the next step, the data is processed using SVM. The outlier detection is done during this step. In the third step the actual soft sensor is built. This is performed by applying a Genetic Programming (GP) algorithm. The GP algorithm selects a function from a pool of available functions and trains it to model the output variable using the pre-processed input data. The soft sensor is a set of analytical functions that map the input space to the target variable space. The proposed approach was applied to several real-life problems, e.g. the interface level estimation in an organic process in [97].

The work of Chen et al. [27] was already mentioned. The soft sensor presented in this work is a grey-box model of a model-driven first principle model and a data-driven RBFN is used to model the non-linear reaction rates. This model is then incorporated into the mass-balance model of a stirred-tank bioreactor. The performance of the proposed hybrid soft sensor is illustrated on an experimental case-study dealing with single microbial population.

A non-traditional approach to soft sensing is presented in [148]. There is an *Intelligent soft sensor* presented in this publication. It is a large system consisting of a symbolic rule-based part, numerical part and a graphical part. This allows the integration of quantitative as well as qualitative knowledge into the model. The three parts are merged by a meta-system. The system is developed for a batch digester quality control support of a sulphite pulping system.

Gonzalez et al. discuss the performance of an ARMAX stepwise regression, Takagi and Sugeno, fuzzy combinational, PLS, wavelet-based and MLP models in [72]. All these models are applied to a rougher flotation bank modelling. The model input are both the process measurements and the combined features. The combined features are built using process knowledge and represent meaningful process descriptors. Apart from this contribution, a novel two-level approach for outlier detection combining PCA capabilities and Scheffé's test is provided. After the application to the modelling of copper concentration grade, the authors conclude that the dynamic PLS as well as the MLP and wavelet-based models are providing best performance.

## A.2 Process monitoring and fault detection applications

Nomikos and MacGregor published a pioneering work on the application of PCA-based techniques for batch and semi-batch process monitoring in [132]. In this work they provide a thorough analysis of the applicability of Statistical Process Control (SPC) charts to the batch process online monitoring. The monitoring of new batches is based on the comparison of their PCA-space representation to reference curves. The reference curves are based on a set of past *good* processes. Based on the reference batches, there is also a possibility to calculate the control limits. In the case of the violation of these limits an alarm is raised and an analysis of the process fault can be done. The presented technique is evaluated on an industrial polymerisation batch process.

Dealing with the application aspects of the PCA and related methods to the process industry problems is [117]. The focus is put on the development of a Recursive PCA (RPCA) approach targeting adaptive process monitoring. Within this framework it has also been shown that the method can deal with outliers, missing values and delayed measurements. The authors presented an effective approach for the update of the correlation matrices as well as two algorithms for the incremental update of the PCA base using the old PCA structure. Additionally, a review of the most common techniques for the selection of the number of principal components is also presented. Based on the review, a new technique for recursive selection of the number of principal components is shown. For the purpose of the adaptive process monitoring, it is necessary to update the confidence limits of the model with the new incoming data, therefore the authors also define a monitoring scheme, which detects and handles data outliers, missing values and process faults

before updating the model. Finally, the proposed monitoring scheme is applied to a rapid thermal annealing process monitoring.

The Model-Based PCA (MBPCA) method is applied to the fault detection of an ethylene compressor in [151]. The detection system is based on a first principle model of the process, which makes the method applicable only to this specific process.

A process monitoring soft sensor using an adaptive version of the PCA (Fast Moving Window PCA-FMWPCA) was published in [188]. The adaptivity of the model is achieved by updating the data structures necessary for the PCA calculation using a novel moving window technique. This technique updates the PCA base (i.e. removes the oldest data sample and adds the new, current, one) in a single step, which makes this technique computationally efficient. Additionally, an N-step-ahead process monitoring approach is presented, which increases the immunity towards faulty data. The effectiveness of the described algorithm is demonstrated using a simulated Fluid Catalytic Cracking unit process.

In [3], an application of PCA and PLS to batch process monitoring is presented. The proposed procedure is split into two steps, the first is applying the PCA to manually explore the data space and to identify reference or *good* batches, which are used in the second stage to develop the PLS model. Having a PLS model of this reference batch, one can compare the new incoming process data (test data) to this model. If there is a deviation between the new data and the reference model data, an analysis of the PLS scores provides information about the variable(s) causing the deviation. The authors are also planning to develop a database of typical process faults and to use an expert system for automatic process fault identification.

The applicability of the PLS, namely the Multi-way PLS (MPLS) algorithm to modelling of batch process quality variables as well as process monitoring and control was presented in [205]. The studied process is a simulated penicillin production fermentation batch process. The quality variable prediction is done using a standard PLS regression model. The process monitoring is carried out using the SPE and $T^2$-statistics of the model.

An alternative approach to process monitoring and process fault detection is discussed in [78]. The presented method is a three-step approach to process monitoring. The first step is called *Pre-analysis* and at this stage a number of clusters in the process data are manually estimated using 2D and 3D PCA-score plots. Using the estimated number of clusters, the data is partitioned by the k-means algorithm. In the second step, the data are visualised after transforming them using the Fisher Discriminant Analysis (FDA). The authors prefer to use the FDA instead of the PCA due to the discrimination abilities of the FDA. Within this step, the normal and faulty process states are annotated. The final step is then the calculation of the fault directions for the separate fault classes using the pairwise FDA. The calculated fault direction provides information about the source of the particular process fault. The algorithm is applied to a simulated as well as to an industrial process.

The identification of batch process end points is dealt with in [125]. The applied technique is the MPLS. The devised technique proves to be very effective and can thus be implemented for the real-time batch process monitoring.

A set of practical applications of process monitoring and quality prediction using a Self Organizing Map (SOM) was published in [2]. In this work, SOMs have been found useful for the monitoring of a continuous pulp digester. Before feeding the data into the SOM model, they have been manually pre-processed using process knowledge. Another application presented in the work is the quality prediction of steel production based on the concentration of the input elements and some process parameters. The last application of SOMs presented in the work is the analysis of the data from paper and pulp industry.

A complex soft sensor for process fault detection and identification has been presented in [201]. The soft sensor is based on an MLP and is applied to the detection of three typical faults in an FCC reactor. The MLP is fed with input from different sources. One source of input is a model-driven soft sensor. This sensor predicts the catalyst circulation rate based on the energy balance equation within the FCC reactor. The output of the soft sensor is then mapped to trends of the catalyst circulation rate, e.g. stable, increasing, etc. The trends are then provided to the MLP. The other inputs to the MLP are trends of directly measurable process variables like the reactor temperature, reactor feed flow rate, etc., which are determined using the wavelet transformation. The developed approach works well for the given process but because of the involvement of the process specific FPM, it is not applicable to any other processes.

In a recent publication [100], a complex soft sensor for the detection and isolation of process faults is devised, which is based on PCA, RBFN and SOM. The soft sensor is developed in the framework of an ethylene cracking process. The authors demonstrate improved accuracy of the system after including calculated variables, which are built using process knowledge. The final soft sensor achieves high performance and is included into the model predictive control of the process.

## A.3 Sensor fault detection and reconstruction applications

Process and sensor faults are detected and handled using the PCA in [51] and [52]. The faults are detected in the PCA residual space. This has the advantage that one can, on one hand, identify the sensor or process faults effectively and on the other hand, by projecting the fault state to the original space one can also find which particular sensor or set of sensors are responsible for the fault. By manipulating the PCA residual space one can also achieve a reconstruction of the fault. The work also defines conditions of the fault detectability, identifiability and reconstructability. For the task of process fault detection there is a need for the description of the *fault direction*, which requires the input of process knowledge to the soft sensor. For the sensor fault detection there is no need for such a knowledge. The proposed approach is again evaluated in terms of an industrial boiler continuous process.

In [114], the previous approach was extended to dynamic processes. The extension to dynamic processes is achieved by using the Time-Lagged PCA (TLPCA) instead of the traditional static PCA method. Although there is a need to remove low auto- and cross-correlated variables from the data set, the presented method is claimed to be suitable for highly dynamic processes, which is demonstrated on one simulated and one industrial data set.

Another PCA-based sensor fault detection and diagnosis soft sensor was published in [186]. The soft sensor uses the Q-statistics to detect faults and the sensors responsible for them. The underlining process is a centrifugal chiller system. The same authors published another fault detection soft sensor in [187], this time monitoring an Air Handling Unit (AHU). In order to deal with the non-linearity of the process the model is split into two separate models. Additionally, the model is extended using a simple expert system, which handles the signals from the two PCA sub-models.

# Appendix B

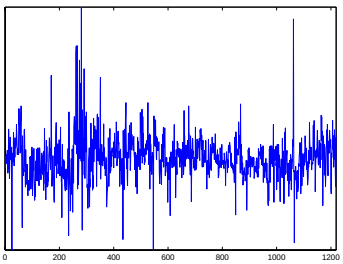
# Data sets

This appendix provides details of the used data set. All of the data sets were kindly provided by Evonik Degussa GmbH and consist of real measurement from Evonik's chemical processes. Due to confidentiality reasons, the displayed variables are anonymised, i.e. mapped to the range $[0, 1]$ and no further details than those provided in the following description can be presented.

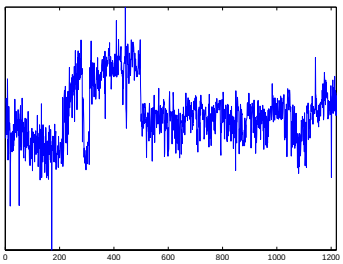
## B.1 Industrial drier

The target values of this data set are laboratory measurements of residual humidity of the process product. The measurement of the target values is done manually in laboratories. Due to the high cost of the measurements it is done only every four hours. The data set consists of 19 input variables, most of them being temperatures, pressures and humidities measured within the processing plant. The data set consists of 1219 data samples covering almost seven months of the operation of the process. It consists of raw unprocessed data as it was recorded by the process information and measurement system. As a result of this, many of the input variables present common issues of industrial data like measurement noise, missing value or data outliers.

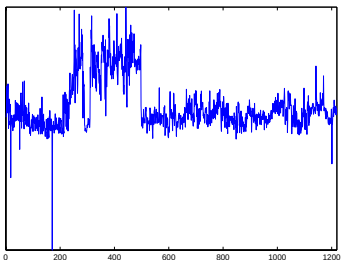
Figure B.1 shows the target variable (Figure B.1i) and the 19 input variables (Figures B.1ii-B.1xx) of the data set. For this data, there was no manual data pre-processing done.



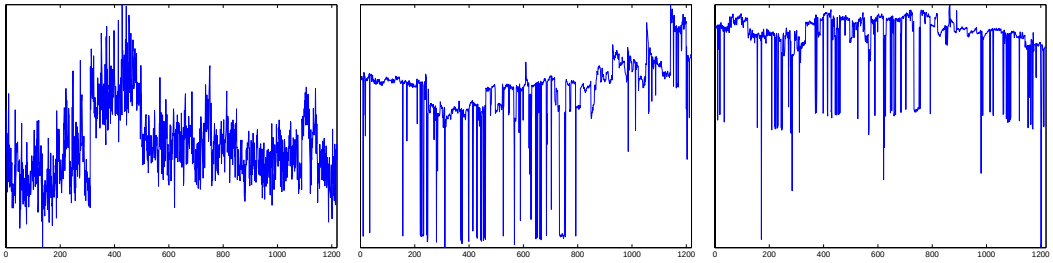

(i) Target variable



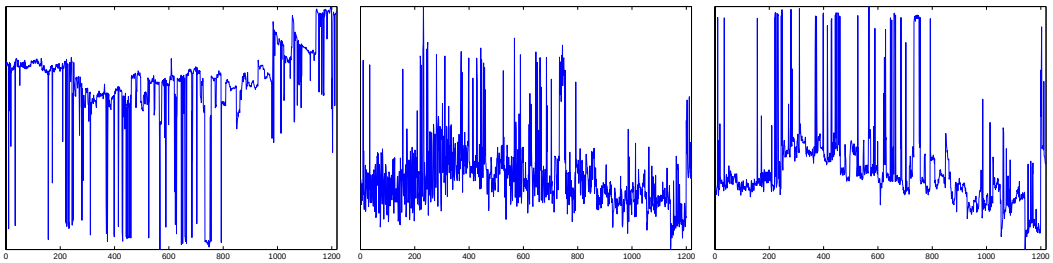

(ii) 1. input variable



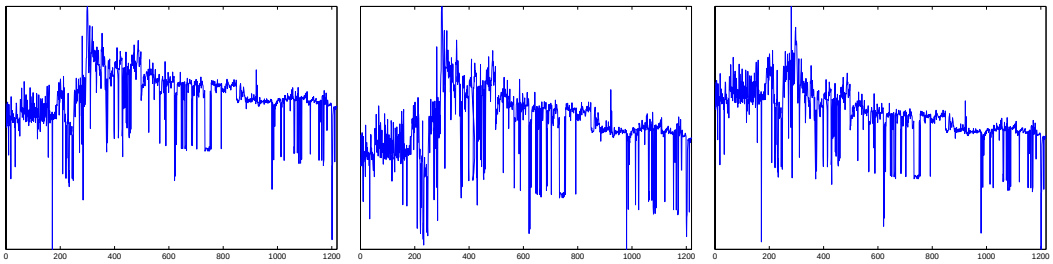

(iii) 2. input variable



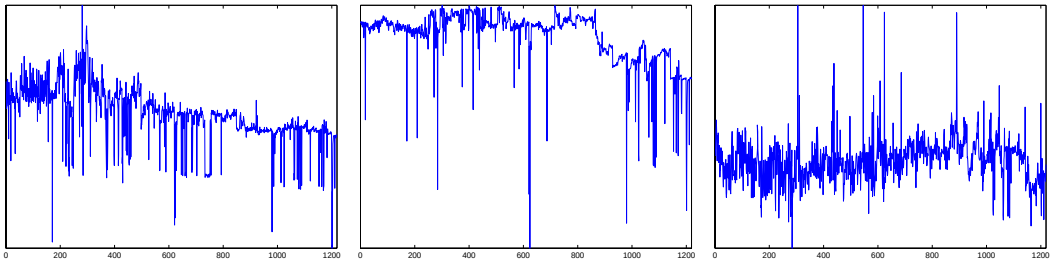
(iv) 3. input variable (v) 4. input variable (vi) 5. input variable



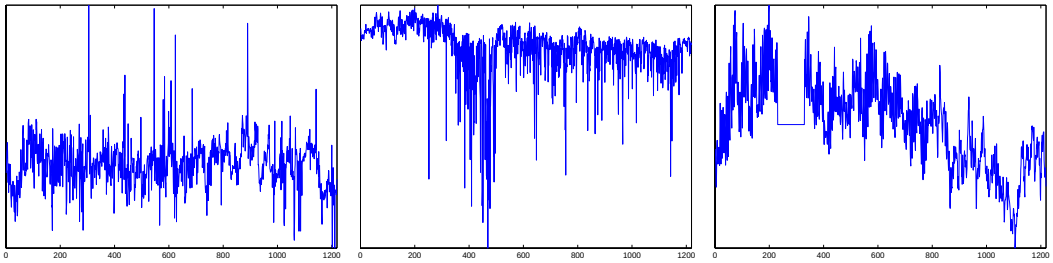
(vii) 6. input variable (viii) 7. input variable (ix) 8. input variable



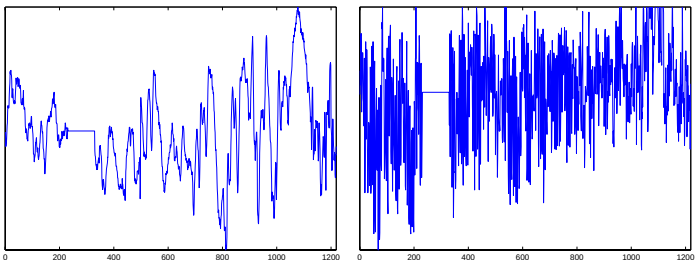
(x) 9. input variable (xi) 10. input variable (xii) 11. input variable



(xiii) 12. input variable (xiv) 13. input variable (xv) 14. input variable



(xvi) 15. input variable (xvii) 16. input variable (xviii) 17. input variable



(xix) 18. input variable (xx) 19. input variable

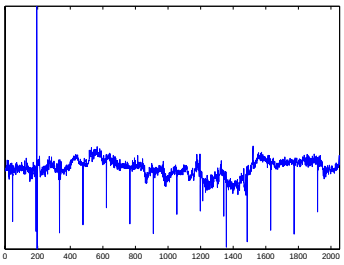
Figure B.1: Industrial drier data set

## B.2 Thermal oxidiser

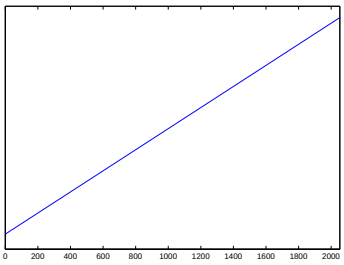
This process industry data set deals with the prediction of exhaust gas concentration of an industrial process. The task is to predict the concentrations of $NO_x$ in the exhaust gases. The data set consists of 39 input features (i.e. hard sensor measurements). The input features are physical values like concentrations, flows, pressures and temperatures measured during the operation of the plant. The data set consists of 2053 samples.

The only manual pre-processing done for this data set is the manual removal of variables 1, 2, 4, 39, which correspond either to the time stamps (variable 1) or to variable severely affected by missing values. This type of pre-processing belongs to the Stage 1: Manual data pre-processing in Figure 4.3.

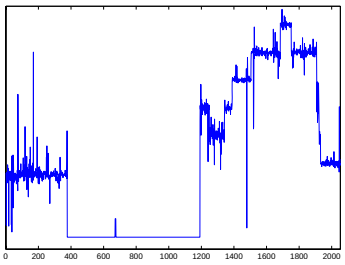
The target variable of this data set is shown in Figure B.2i and the input variables are shown in Figures B.2ii-B.2xl.



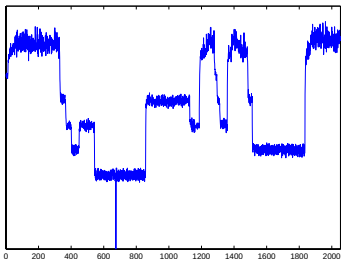

(i) Target variable



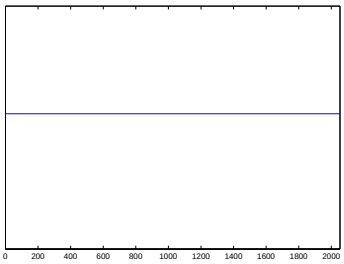

(ii) 1. input variable



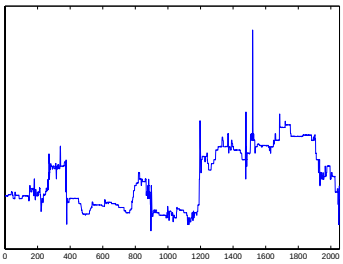

(iii) 2. input variable



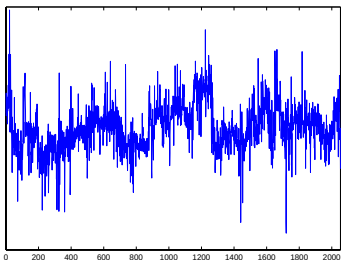

(iv) 3. input variable



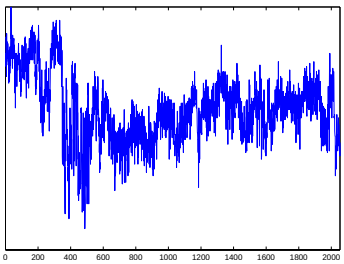

(v) 4. input variable



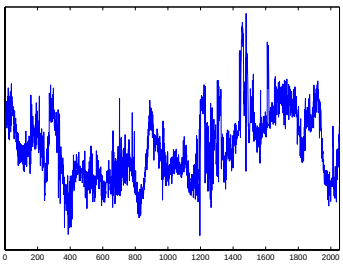

(vi) 5. input variable



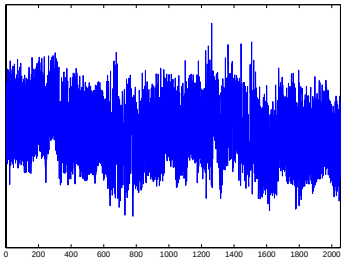

(vii) 6. input variable



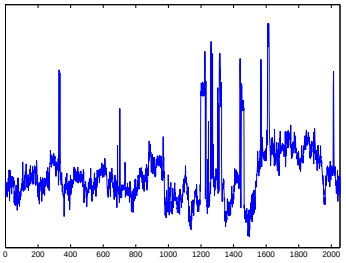

(viii) 7. input variable



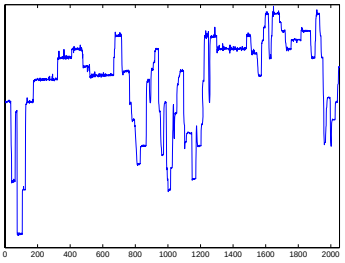

(ix) 8. input variable



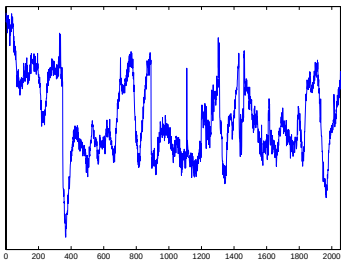

(x) 9. input variable



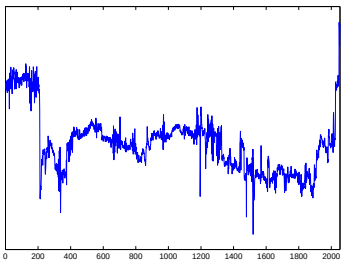

(xi) 10. input variable



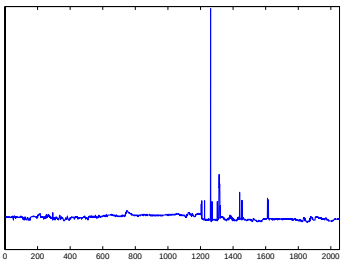

(xii) 11. input variable



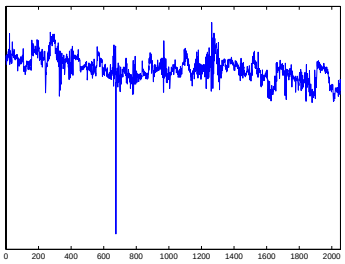

(xiii) 12. input variable



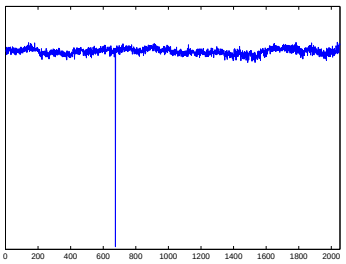

(xiv) 13. input variable



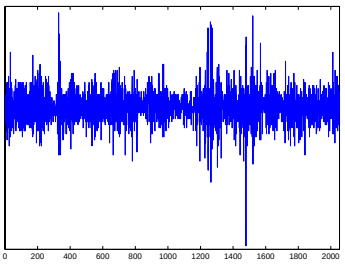

(xv) 14. input variable



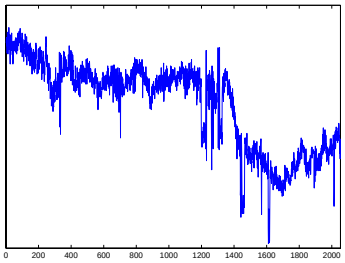

(xvi) 15. input variable



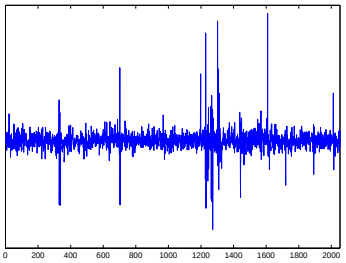

(xvii) 16. input variable



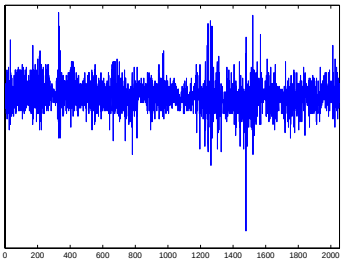

(xviii) 17. input variable



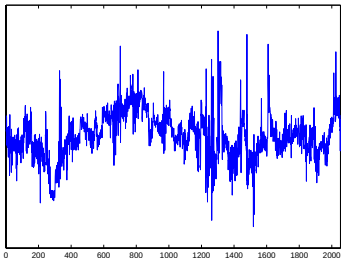

(xix) 18. input variable



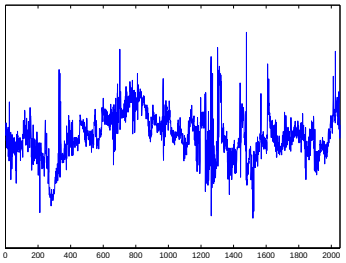

(xx) 19. input variable



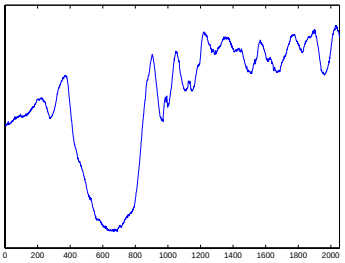

(xxi) 20. input variable



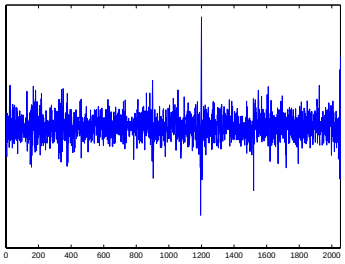

(xxii) 21. input variable



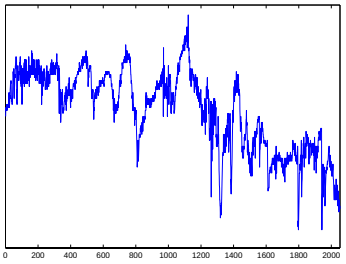

(xxiii) 22. input variable



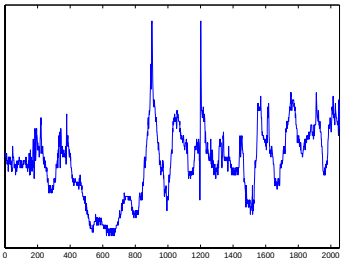

(xxiv) 23. input variable



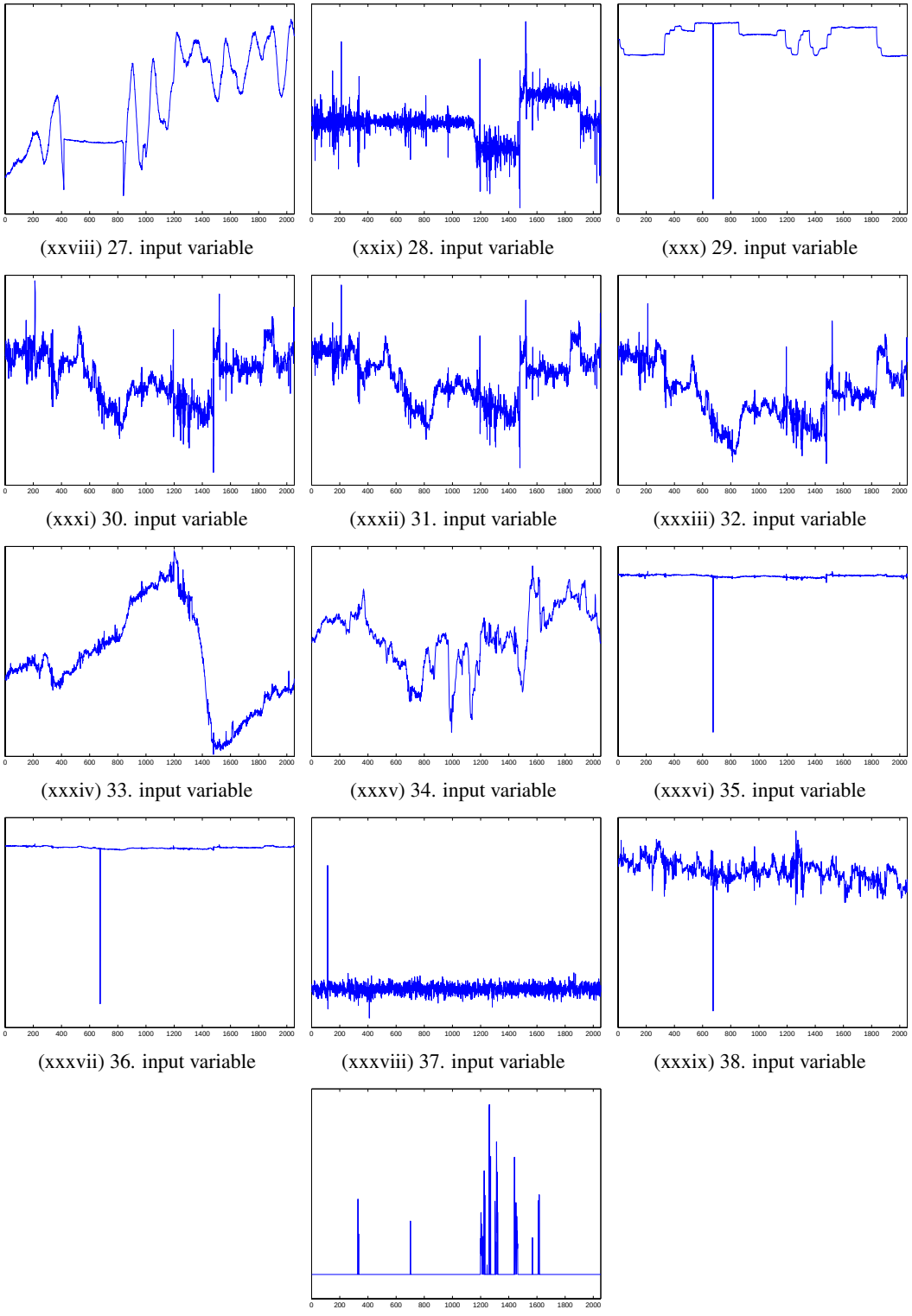

(xxv) 24. input variable




(xxvi) 25. input variable




(xxvii) 26. input variable




(xxviii) 27. input variable (xxix) 28. input variable (xxx) 29. input variable

(xxxi) 30. input variable (xxxii) 31. input variable (xxxiii) 32. input variable

(xxxiv) 33. input variable (xxxv) 34. input variable (xxxvi) 35. input variable

(xxxvii) 36. input variable (xxxviii) 37. input variable (xxxix) 38. input variable

(xl) 39. input variable

Figure B.2: Thermal oxidiser data set

## B.3 Catalyst activation

This data set was used for the NiSIS 2006 competition[1]. However, for the competition the data were handled in a way that is not compatible to the data handling in this work and thus the results cannot be compared.

The reactor to be modelled consists of some 1000 tubes filled with catalyst, used to oxidize a gaseous feed (ethane is taken as example). It is cooled with a coolant supposed to be at constant temperature. The description of the reaction speed is taken from literature and depends strongly non-linearly from temperature. Its exothermal reaction is counteracted by the cooling and leads to a temperature maximum somewhere along the length of the tube. As the catalyst decays, this becomes less pronounced and moves further downstream. The catalyst activity usually decays within some time to zero, a year is taken as example here. The process to be modelled takes input from other, larger processes, so that the feed will vary over the days. The operating personal reacts to this by choosing appropriate operating conditions. The catalyst decay is however much slower than these effects. The process is equipped with measurements to log all the variations of the feed and the operating conditions. In addition, there are measurements showing some concentrations, flows and a lot of temperatures along the length of a characteristic tube to identify the processes state. The reactor and the product are shown in Figure B.3

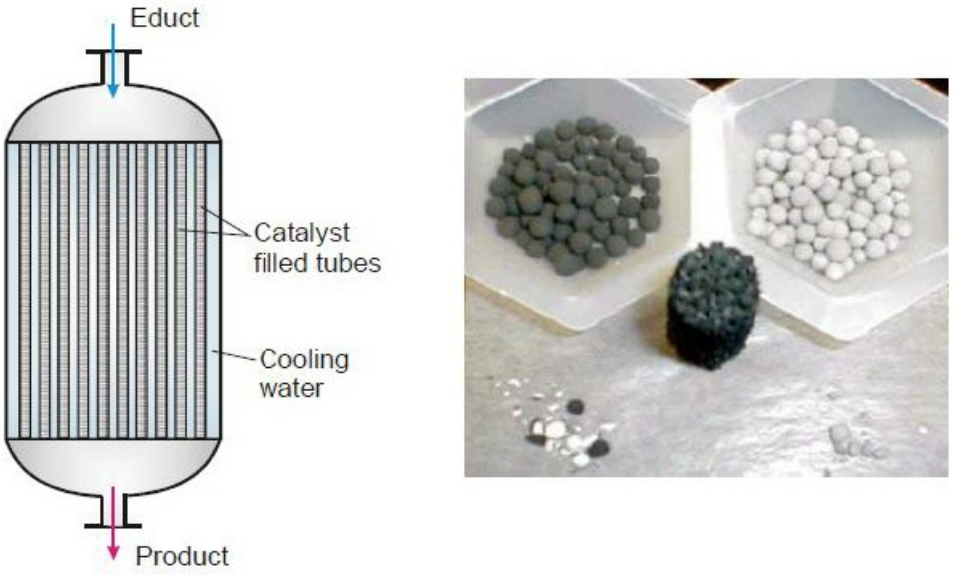



Figure B.3: The reactor and product related to the catalyst activation data

The task was to predict the activity of a catalyst in a multi-tube reactor. The input data are 14 sensor measurements like flows, concentrations and temperatures, from a real process together with one variable with the timestamps of the measurements. The target variable is the simulated activity of the catalyst inside the reactor. The data set covers one year of operation of the process plant. Many of the variables show high co-linearity of the features and high amount of outliers, which can be found in as many as $80\%$ of the variables.

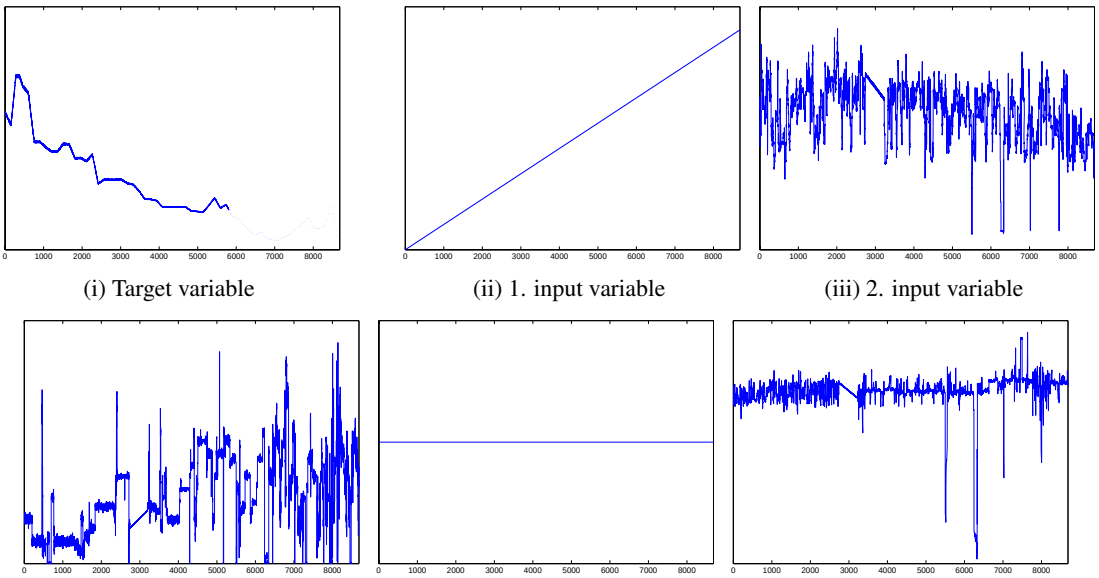
Table B.1 describes the physical values represented by the variables in this data set.

[1] http://www.nisis.risk-technologies.com/filedown.aspx?file=125

| | |
|---|---|
| 1. variable | time |
| 2. variable | measured flow of air |
| 3. variable | measured flow of combustible gas |
| 4. variable | measured concentration of combustible component in the combustible gass |
| 5. variable | total feed temperature |
| 6. variable | cooling temperature |
| 7. variable | Temperature at length 1/20 of reactor length |
| 8. variable | Temperature at length 2/20 of reactor length |
| 9. variable | Temperature at length 4/20 of reactor length |
| 10. variable | Temperature at length 7/20 of reactor length |
| 11. variable | Temperature at length 11/20 of reactor length |
| 12. variable | Temperature at length 16/20 of reactor length |
| 13. variable | Temperature at length 20/20 of reactor length |
| 14. variable | Product concentration of oxygen |
| 15. variable | Product concentration of combustible component |

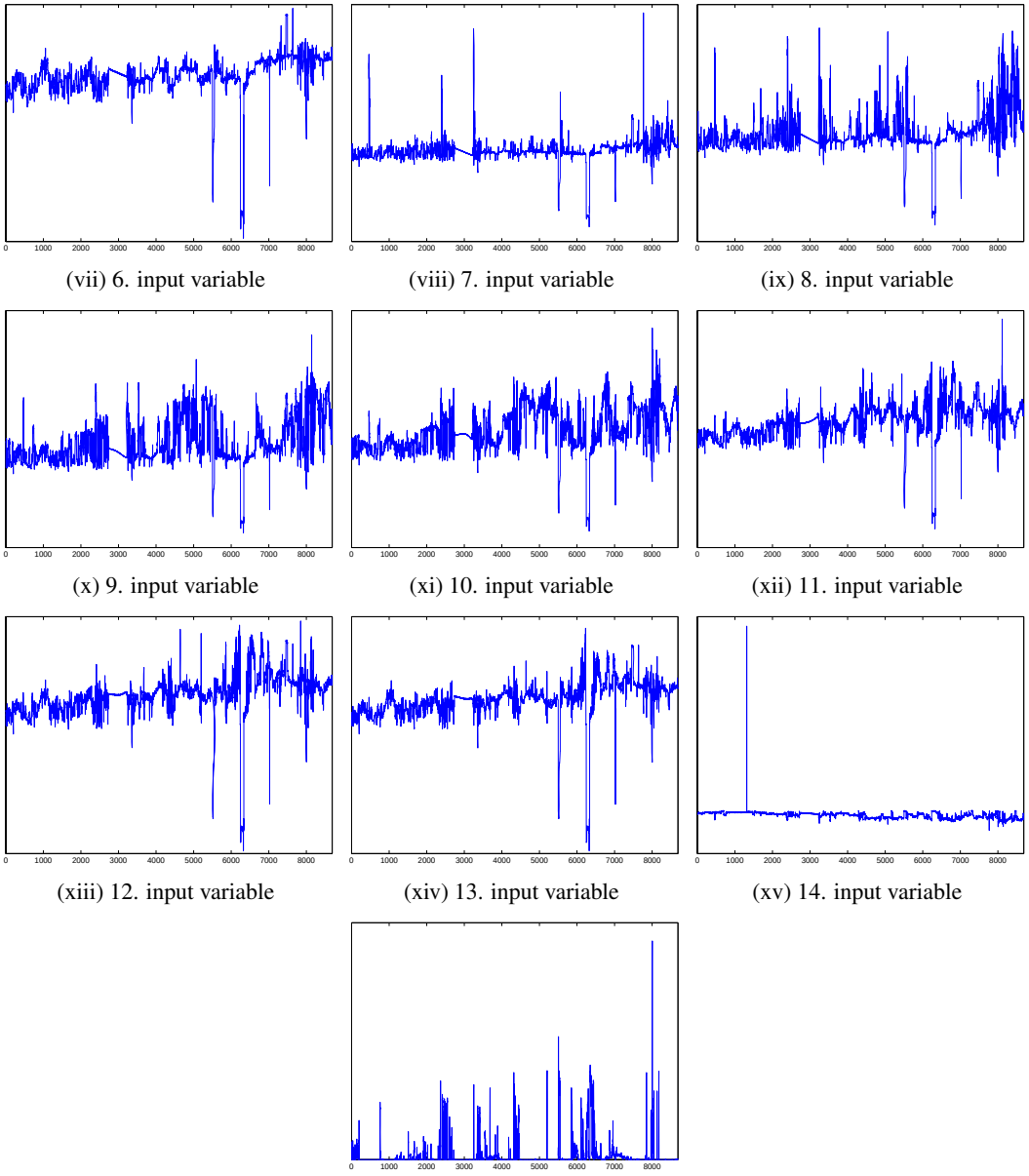
Table B.1: The measurements in the catalyst activation data set

In order to increase the feasibility of the data for the computational processing, the data were down sampled and only every 10th date point was used for the historical and on-line data. Additionally, the samples without any target value (see Figure B.4i) were also skipped as for these values the prediction error cannot be calculated.



(i) Target variable (ii) 1. input variable (iii) 2. input variable

(iv) 3. input variable (v) 4. input variable (vi) 5. input variable




(vii) 6. input variable (viii) 7. input variable (ix) 8. input variable

(x) 9. input variable (xi) 10. input variable (xii) 11. input variable

(xiii) 12. input variable (xiv) 13. input variable (xv) 14. input variable

(xvi) 15. input variable

Figure B.3: Catalyst activation data set

# Bibliography


[1] D. Aha. Lazy learning. *Artificial Intelligence Review*, 11(1-5):7–10, 1997.

[2] E. S. A. Alhoniemi. Process Monitoring and Modeling Using the Self-Organizing Map. *Integrated Computer-Aided Engineering*, 6(1):3–14, 1999.

[3] M. Amazouz and R. Pantea. Use of multivariate data analysis for lumber drying process monitoring and fault detection. In S. F. Crone, S. Lessmann, and R. Stahlbock, editors, *Proceedings of the International Conference on Data Mining (ICDM)*, pages 329–332, 2006.

[4] M. J. Arauzo-Bravo, J. M. Cano-Izquierdo, E. Gomez-Sanchez, M. J. Lopez-Nieto, Y. A. Dimitriadis, and J. Lopez-Coronado. Automatization of a penicillin production process with soft sensors and an adaptive controller based on neuro fuzzy systems. *Control Engineering Practice*, 12(9):1073–1090, 2004.

[5] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally Weighted Learning. *Artificial Intelligence Review*, 11(1):11–73, 1997.

[6] G. Bastin and D. Dochain. *On-line estimation and adaptive control of bioreactors*. Elsevier, Amsterdam, 1990.

[7] E. Bauer and R. O. N. Kohavi. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine Learning*, 36:105–139, 1999.

[8] J. Baxter. Learning Model Bias. *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, (8):169–175, 1996.

[9] J. Baxter. Theoretical models of learning to learn. In S. Mitchell, T.; Thrun, editor, *Learning to Learn*, pages 71–94. Kluwer, Boston, 1998.

[10] H. Bensusan, C. Giraud-Carrier, and C. Kennedy. A higher-order approach to meta-learning. In *Proceedings of the ECML2000 workshop on Meta-Learning*, pages 109–117, 2000.

[11] H. N. Bensusan. *Automatic Bias Learning: An Inquiry Into the Inductive Basis of Induction*. PhD thesis, 1999.

[12] A. Bifet and R. Gavalda. Learning from Time-Changing Data with Adaptive Windowing. In *Proceedings of the SIAM International Conference on Data Mining (SDM07)*. Technical report, Universitat Politecnica de Catalunya, 2006. Available from www. lsi. upc. edu/abifet, 2006.

[13] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, USA, 1995.

[14] D. Bonne and S. B. Jorgensen. Data-Driven Modeling of Batch Processes. In *Proceedings of 7th International Symposium on Advanced Control of Chemical Processes*, 2004.

[15] L. Bottou and V. Vapnik. Local learning algorithms. *Neural Computation*, 4(6):888–900, 1992.

[16] P. Brazdil. Data transformation and model selection by experimentation and meta-learning. Technical Report CSR-98-02, 1995.

[17] L. Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–140, 1996.

[18] G. Brown. *Diversity in Neural Network Ensembles*. Phd thesis, 2004.

[19] G. Brown, J. Wyatt, R. Harris, and X. Yao. Diversity creation methods: a survey and categorisation. *Information Fusion*, 6(1):5–20, 2005.

[20] G. Brown, J. L. Wyatt, and P. Tino. Managing Diversity in Regression Ensembles. *The Journal of Machine Learning Research*, 6:1621–1650, 2005.

[21] G. A. Carpenter and S. Grossberg. Adaptive resonance theory (ART). In M. A. Arbib, editor, *The handbook of brain theory and neural networks table of contents*, pages 79–82. MIT Press, Cambridge, 1998.

[22] A. Casali, G. Gonzalez, F. Torres, G. Vallebuona, L. Castelli, and P. Gimenez. Particle size distribution soft-sensor for a grinding circuit. *Powder Technology*, 99(1):15–21, 1998.

[23] C. Castiello, G. Castellano, and A. M. Fanelli. Meta-data: Characterization of Input Features for Meta-learning. *LECTURE NOTES IN COMPUTER SCIENCE*, 3558:457–468, 2005.

[24] C. Castiello and A. M. Fanelli. Mechanisms for Inductive Learning: From Base-learning to Meta-learning. In *Proceedings of IASTED International Conference on Artificial Intelligence and Application (AIA2004)*, pages 276–280, 2004.

[25] M. Champagne, M. Dudzic, T. Inc, and Q. Temiscaming. Industrial use of multivariate statistical analysis for process monitoring and control. In *Proceedings of the 2002 American control conference*, volume 1, 2002.

[26] A. Chandra and X. Yao. Evolving hybrid ensembles of learning machines for better generalisation. *Neurocomputing*, 69(7-9):686–700, 2006.

[27] L. Chen, O. Bernard, G. Bastin, and P. Angelov. Hybrid modelling of biotechnological processes using neural networks. *Control Engineering Practice*, 8(7):821–827, 2000.

[28] L. Z. Chen, S. K. Nguang, X. M. Li, and X. D. Chen. Soft sensors for on-line biomass measurements. *Bioprocess and Biosystems Engineering*, 26(3):191–195, 2004.

[29] X. Chen, F. Gao, and G. Chen. A soft-sensor development for melt-flow-length measurement during injection mold filling. *Materials Science and Engineering A*, 384(1-2):245–254, 2004.

[30] S. W. Choi, E. B. Martin, A. J. Morris, and I. B. Lee. Adaptive Multivariate Statistical Process Control for Monitoring Time-Varying Processes. *Industrial and Engineering Chemistry Research*, 45:3108–3118, 2006.

[31] A. Chruy. Software sensors in bioprocess engineering. *Journal of Biotechnology*, 52(3):193–199, 1997.

[32] F. Chu, Y. Wang, and C. Zaniolo. An adaptive learning approach for noisy data streams. In *Proceedings of the 4th IEEE International Conference on Data Mining*, pages 351–354, 2004.

[33] P. J. Chung and J. F. Bohme. Recursive EM algorithm with adaptive step size. In *Seventh International Symposium on Signal Processing and Its Applications*, volume 2, pages 519–522. IEEE, 2003.

[34] C. Croux and G. Haesbroeck. Principal component analysis based on robust estimators of the covariance or correlation matrix: influence functions and efficiencies. *Biometrika*, 87(3):603–618, 2000.

[35] C. Croux and A. Ruiz-Gazen. High breakdown estimators for principal components: the projection-pursuit approach revisited. *Journal of Multivariate Analysis*, 95(1):206–226, 2005.

[36] L. Davies and U. Gather. The Identification of Multiple Outliers. *Journal of the American Statistical Association*, 88(423):782–792, 1993.

[37] B. S. Dayal and J. F. MacGregor. Recursive exponentially weighted PLS and its applications to adaptive control and prediction. *Journal of Process Control*, 7(3):169–179, 1997.

[38] S. De Wolf, R. L. E. Cuypers, L. C. Zullo, B. J. Vos, and B. J. Bax. Model predictive control of a slurry polymerisation reactor. *Computers and Chemical Engineering*, 20:955–961, 1996.

[39] K. Desai, Y. Badhe, S. S. Tambe, and B. D. Kulkarni. Soft-sensor development for fed-batch bioreactors using support vector regression. *Biochemical Engineering Journal*, 27(3):225–239, 2006.

[40] D. Devogelaere, M. Rijckaert, O. G. Leon, and G. C. Lemus. Application of feedforward neural networks for soft sensors in the sugar industry. In *Proceedings of VII Brazilian Symposium on Neural Networks*, pages 2–6, 2002.

[41] F. Ding and T. Chen. Modeling and identification for multirate systems. *Acta Automatica Sinica*, 31(1):105–122, 2005.

[42] D. Dong and T. J. McAvoy. Nonlinear principal component analysis–based on principal curves and neural networks. *Computers and Chemical Engineering*, 20(1):65–78, 1996.

[43] D. Dong, T. J. McAvoy, and L. J. Chang. Emission monitoring using multivariate soft sensors. In *Proceedings of the American Control Conference*, volume 1, 1995.

[44] Y. Dote and S. J. Ovaska. Industrial Applications of Soft Computing: A Review. *Proceedings of the IEEE*, 89(9):1243–1265, 2001.

[45] F. J. Doyle. Nonlinear inferential control for process applications. *Journal Process Control*, 8(5-6):339–353, 1998.

[46] H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. In *Advances in Neural Information Processing Systems*, pages 155–161. Morgan Kaufmann Publishers, 1997.

[47] H. Drucker, C. Cortes, L. D. Jackel, Y. LeCun, and V. Vapnik. Boosting and Other Ensemble Methods. *Neural Computation*, 6(6):1289–1301, 1994.

[48] R. O. Duda, P. E. Hart, and D. G. Stork. *Patten Classification*. John Wiley and Sons, New York, 2001.

[49] N. Duffy and D. Helmbold. Boosting Methods for Regression. *Machine Learning*, 47(2):153–200, 2002.

[50] R. Dunia, J. Qin, T. F. Edgar, and T. J. McAvoy. Sensor fault identification and reconstruction using principal component analysis. In *Proceedings of the 13th Triennial IFAC World Congress*, pages 259–264, 1996.

[51] R. Dunia and S. J. Qin. Joint diagnosis of process and sensor faults using principal component analysis. *Control Engineering Practice*, 6(4):457–469, 1998.

[52] R. Dunia and S. J. Qin. Subspace Approach to Multidimensional Identification and Reconstruction. *AIChE Journal*, 44(8):1813, 1998.

[53] M. Fellner, A. Delgado, and T. Becker. Functional nodes in dynamic neural networks for bioprocess modelling. *Bioprocess and Biosystems Engineering*, 25(5):263–270, 2003.

[54] R. Feng, W. Shen, and H. Shao. A soft sensor modeling approach using support vector machines. In *Proceedings of the 2003 American Control Conference*, volume 5, 2003.

[55] L. Fortuna. *Soft Sensors for Monitoring and Control of Industrial Processes*. Springer Verlag, London, 2007.

[56] L. Fortuna, S. Graziani, and M. G. Xibilia. Soft sensors for product quality monitoring in debutanizer distillation columns. *Control Engineering Practice*, 13(4):499–508, 2005.

[57] E. Frank, M. Hall, and B. Pfahringer. Locally Weighted Naive Bayes. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 249–256, 2003.

[58] R. French. Catastrophic forgetting in connectionist networks: Causes, consequences and solutions. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

[59] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[60] K. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2(3):183–192, 1989.

[61] B. Gabrys and A. Bargiela. General Fuzzy Min-Max Neural Network for Clustering and Classification. *IEEE Transaction on neural networks*, 11(3):769–783, 2000.

[62] B. Gabrys and L. Petrakieva. Combining labelled and unlabelled data in the design of pattern classification systems. *International Journal of Approximate Reasoning*, 35(3):251–273, 2004.

[63] B. Gabrys and D. Ruta. Genetic algorithms in classifier fusion. *Applied Soft Computing*, 6(4):337–347, 2006.

[64] J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with drift detection. In *Proceedings of the 17th Brazilian Symposium on Artificial Intelligence Advances in Artificial Intelligence (SBIA 2004)*, volume 3171, pages 286–295, 2004.

[65] P. Geladi and K. Esbensen. Regression on multivariate images: principal component regression for modeling, prediction and visual diagnostic tools. *Journal of Chemometrics*, 5(2):97–111, 1991.

[66] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

[67] C. Giraud-Carrier. Beyond predictive accuracy: what? In *Proceedings of the ECML-98 Workshop on Upgrading Learning to Meta-Level*, pages 78–85, 1998.

[68] C. Giraud-Carrier and F. Provost. Toward a Justification of Meta-learning: Is the No Free Lunch Theorem a Show-stopper? In *Proceedings of the ICML 2005- Workshop on Meta-learning*, Bonn, 2005.

[69] C. Giraud-Carrier, R. Vilalta, and P. Brazdil. Introduction to the Special Issue on Meta-Learning. *Machine Learning*, 54(3):187–193, 2004.

[70] E. Gomez, H. Unbehauen, P. Kortmann, and S. Peters. Fault detection and diagnosis with the help of fuzzy-logic and with application to a laboratory turbogenerator. In *Proceedings of the 13th IFAC World Congress*, volume N, pages 175–180, 1996.

[71] G. D. Gonzalez. Soft sensors for processing plants. *Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials (IPMM'99)*, 1, 1999.

[72] G. D. Gonzalez, M. Orchard, J. L. Cerda, A. Casali, and G. Vallebuona. Local models for soft-sensors in a rougher flotation bank. *Minerals Engineering*, 16(5):441–453, 2003.
[73] W. S. Gosset. The probable error of a mean. *Biometrika*, 6(1):1–25, 1908.
[74] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3(7-8):1157–1182, 2003.
[75] C. Han and Y. H. Lee. Intelligent integrated plant operation system for Six Sigma. *Annual Reviews in Control*, 26(1):27–43, 2002.
[76] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.
[77] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.
[78] Q. P. He, S. J. Qin, and J. Wang. A new fault diagnosis method using fault directions in Fisher discriminant analysis. *AIChE Journal*, 51(2):555–571, 2005.
[79] V. Hodge and J. Austin. A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.
[80] H. Hotelling. The Generalization of Student's Ratio. *The Annals of Mathematical Statistics*, 2(3):360–378, 1931.
[81] D. H. Hubel. The visual cortex of the brain. *Sci Am*, 209:54–62, 1963.
[82] P. Huber and E. Ronchetti. *Robust statistics*. Wiley-Blackwell, 2009.
[83] J. E. Jackson and G. S. Mudholkar. Control Procedures for Residuals Associated with Principal Component Analysis. *Technometrics*, 21(3):341–349, 1979.
[84] R. Jacobs. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.
[85] S. James, R. Legge, and H. Budman. Comparative study of black-box and hybrid estimation methods in fed-batch fermentation. *Journal of Process Control*, 12(1):113–121, 2002.
[86] J. S. R. Jang, C. T. Sun, and E. Mizutani. *Neuro-fuzzy and soft computing*. Prentice Hall Upper Saddle River, NJ, 1997.
[87] T. Jiang, B. Chen, X. He, and P. Stuart. Application of steady-state detection method based on wavelet transform. *Computers and Chemical Engineering*, 27(4):569–578, 2003.
[88] I. T. Jolliffe. *Principal Component Analysis*. Springer, New York, 2002.
[89] E. Jordaan, A. Kordon, L. Chiang, and G. Smits. Robust Inferential Sensors Based on Ensemble of Predictors Generated by Genetic Programming. In *Proceedings ofInternational Conference on Parallel Problem Solving from Nature*, pages 522–531, Birmingham, UK, 2004.
[90] A. Jos de Assis and R. Maciel Filho. Soft sensors development for on-line bioreactor state estimation. *Computers and Chemical Engineering*, 24(2):1099–1103, 2000.
[91] P. Kadlec and B. Gabrys. Adaptive Local Learning Soft Sensor for Inferential Control Support. In *Proceedings of International Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA'2008)*, Vienna, Austria, 2008. IEEE.
[92] P. Kadlec and B. Gabrys. Gating Artificial Neural Network Based Soft Sensor. In N. T. Nguyen and R. Katarzyniak, editors, *New Challenges in Applied Intelligence Technologies*, volume 134 of *Studies in Computational Intelligence*, pages 193–202. Springer-Verlag, 2008.
[93] P. Kadlec and B. Gabrys. Learnt Topology Gating Artificial Neural Network. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 2605–2612, Hong Kong, 2008. IEEE.
[94] P. Kadlec and B. Gabrys. Soft Sensor based on adaptive local learning. In M. K. Coghill, N. Kasabov, and George, editors, *Proceedings of the International Conference On Neural Information Processing*, volume 5506 of *Lecture Notes in Computer Science*, pages 1172–1179, Auckland, New Zealand, 2008. Springer.
[95] P. Kadlec and B. Gabrys. Architecture for development of adaptive on-line prediction models. *Memetic Computing*, 1(4):241–269, 2009.
[96] P. Kadlec, B. Gabrys, and S. Strandt. Data-driven Soft Sensor in the process industry. *Computers and Chemical Engineering*, 33(4):795–814, 2009.
[97] A. Kalos, A. Kordon, G. Smits, and S. Werkmeister. Hybrid Model Development Methodology for Industrial Soft Sensors. In *Proceedings of the American Control Conference*, pages 5417–5422, 2003.

[98] A. Kalousis, J. Gama, and M. Hilario. On Data and Algorithms: Understanding Inductive Performance. *Machine Learning*, 54(3):275–312, 2004.

[99] A. Kalousis and M. Hilario. Model Selection via Meta-Learning: A Comparative Study. *International Journal on Artificial Intelligence Tools*, 10(4):525–554, 2001.

[100] P. Kampjarvi, M. Sourander, T. Komulainen, N. Vatanski, M. Nikus, and S. L. Jms-Jounela. Fault detection and isolation of an on-line analyzer for an ethylene cracking process. *Control Engineering Practice*, 16(1):1–13, 2008.

[101] R. D. King, C. Feng, and A. Sutherland. StatLog: Comparison of Classification Algorithms on Large Real-World Problems. *Applied Artificial Intelligence*, 9(3):289–333, 1995.

[102] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[103] S. Klanke, S. Vijayakumar, and S. Schaal. A Library for Locally Weighted Projection Regression. *The Journal of Machine Learning Research*, 9:623–626, 2008.

[104] R. Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, 8(3):281–300, 2004.

[105] R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, 2000.

[106] T. Kohonen. *Self-organizing maps*. Springer-Verlag, New Jersey, 1997.

[107] A. Kordon, G. Smits, E. Jordaan, E. Rightor, D. C. Co, and T. X. Freeport. Robust soft sensors based on integration of genetic programming, analytical neural networks, and support vector machines. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02)*, volume 1, 2002.

[108] A. K. Kordon. Hybrid intelligent systems for industrial data analysis. *International Journal of Intelligent Systems*, 19(4):367–383, 2004.

[109] A. K. Kordon. Application issues of industrial soft computing systems. *Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS 2005)*, pages 110–115, 2005.

[110] A. K. Kordon, A. N. Kalos, F. A. Castillo, M. E. Kotanchek, E. M. Jordaan, and G. F. Smits. Competitive Advantages of Evolutionary Computation for Industrial Applications. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, volume 1, 2005.

[111] A. Krogh and J. Vedelsby. Neural network ensembles, cross validation and active learning. *Advances in Neural Information Processing Systems*, (7):231–238, 1995.

[112] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, New Jersey, 2004.

[113] M. M. Lazarescu. Using multiple windows to track concept drift. *Intelligent Data Analysis*, 8(1):29–59, 2004.

[114] C. Lee, S. W. Choi, and I.-B. Lee. Sensor fault identification based on time-lagged PCA in dynamic processes. *Chemometrics and Intelligent Laboratory Systems*, 70(2):165–178, 2004.

[115] G. Li and Z. Chen. Projection-pursuit approach to robust dispersion matrices and principal components: primary theory and Monte Carlo. *Journal of the American Statistical Association*, pages 759–766, 1985.

[116] S. J. Li, X. J. Zhang, and F. Qian. Soft Sensing Modeling via Artificial Neural Network Based on Pso-Alopex. In *Proceedings of 2005 International Conference on Machine Learning and Cybernetics*, volume 7, 2005.

[117] W. Li, H. H. Yue, S. Valle-Cervantes, and S. J. Qin. Recursive PCA for adaptive process monitoring. *Journal of Process Control*, 10(5):471–486, 2000.

[118] B. Lin, B. Recke, J. Knudsen, and S. B. Jrgensen. A Systematic Approach for Soft Sensor Development. *Computers and Chemical Engineering*, 31(5):419–425, 2007.

[119] G. Lindner and R. Studer. AST: Support for Algorithm Selection with a CBR Approach. In *Proceedings of Principles of Data Mining and Knowledge Discovery (Pkdd'99)*, Prague, Czech Republic, 1999.

[120] Y. Liu, X. Yao, and T. Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, 4(4):381, 2000.

[121] J. X. Luo and H. H. Shao. Developing soft sensors using hybrid soft computing methodology: a neurofuzzy system based on rough set theory and genetic algorithms. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 10(1):54–60, 2006.

[122] J. J. Macias and P. X. Zhou. A Method for Predicting Quality of the Crude Oil Distillation. In *Proceedings of the International Symposium on Evolving Fuzzy Systems*, pages 214–220, 2006.

[123] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[124] M. A. Maloof and R. S. Michalski. Selecting Examples for Partial Memory Learning. *Machine Learning*, 41(1):27–52, 2000.

[125] O. Marjanovic, B. Lennox, D. Sandoz, K. Smith, and M. Crofts. Real-time monitoring of an industrial batch process. *Computers and Chemical Engineering*, 30(10-12):1476–1481, 2006.

[126] D. Mease and A. Wyner. Evidence contrary to the statistical view of boosting. *The Journal of Machine Learning Research*, 9:131–156, 2008.

[127] L. A. C. Meleiro and R. M. Finho. A self-tuning adaptive control applied to an industrial large scale ethanol production. *Computers and Chemical Engineering*, 24(2-7):925–930, 2000.

[128] P. H. Menold, R. K. Pearson, and F. Allgower. Online outlier detection and removal. In *Proceedings of the 7th Mediterranean on Control and Automation (MED99)*, pages 1110–1133, Haifa, Israel, 1999.

[129] S. Merikoski, M. Laurikkala, and H. Koivisto. An adaptive neuro-fuzzy inference system as a soft sensor for viscosity in rubber mixing process. In *WSEAS NNA-FSFS-EC 2001*, Puerto de la Cruz, Tenerife, Spain, 2001.

[130] M. F. Moller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):5–25, 1993.

[131] I. T. Nabney. *NETLAB: algorithms for pattern recognition*. Springer Verlag, 2002.

[132] P. Nomikos and J. F. MacGregor. Multivariate SPC Charts for Monitoring Batch Processes. *Technometrics*, 37(1):41–59, 1995.

[133] D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

[134] S. Park and C. Han. A nonlinear soft sensor based on multivariate smoothing procedure for quality estimation in distillation columns. *Computers and Chemical Engineering*, 24(2-7):871–877, 2000.

[135] E. Parzen. On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

[136] R. K. Pearson. Exploring process data. *Journal of Process Control*, 11(2):179–194, 2001.

[137] R. K. Pearson. Outliers in process modeling and identification. *IEEE Transactions on Control Systems Technology*, 10(1):55–63, 2002.

[138] Y. Peng, P. A. Flach, C. Soares, and P. Brazdil. Improved Dataset Characterisation for Meta-learning. *Lecture Notes in Computer Science*, (2534):141–152, 2002.

[139] M. P. Perrone. *Improving Regression Estimation: Averaging Methods for Variance Reduction with Extensions to General Convex Measure Optimization*. PhD thesis, 1993.

[140] M. P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. *Neural Networks for Speech and Image Processing*, pages 126–142, 1993.

[141] B. Pfahringer, H. Bensusan, and C. Giraud-Carrier. Meta-learning by landmarking various learning algorithms. In *Proceedings of the Seventeenth International Conference on Machine Learning*, volume 951, pages 743–750. Morgan Kaufmann, 2000.

[142] V. Prasad, M. Schley, L. P. Russo, and B. Wayne Bequette. Product property and production rate control of styrene polymerization. *Journal of Process Control*, 12(3):353–372, 2002.

[143] R. B. C. Prudencio and T. B. Ludermir. Active Meta-Learning with Uncertainty Sampling and Outlier Detection. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 346–351, Hong Kong, China, 2008. IEEE.

[144] S. J. Qin. Neural networks for intelligent sensors and control - Practical issues and some solutions. *Neural Systems for Control*, pages 213–234, 1997.

[145] S. J. Qin. Recursive PLS algorithms for adaptive data modeling. *Computers and Chemical Engineering*, 22(4-5):503–514, 1998.

[146] S. J. Qin, H. Yue, and R. Dunia. Self-validating inferential sensors with application to air emission monitoring. *Industrial and Engineering Chemistry Research*, 36:1675–1685, 1997.

[147] V. R. Radhakrishnan and A. R. Mohamed. Neural networks for the identification and control of blast furnace hot metal quality. *Journal of Process Control*, 10(6):509–524, 2000.

[148] M. Rao, J. Corbin, and Q. Wang. Soft sensors for quality prediction in batch chemical pulping processes. In *Proceedings of the IEEE International Symposium on Intelligent Control*, pages 150–155, 1993.

[149] S. Riedel and B. Gabrys. Dynamic Pooling for the Combination of Forecasts generated using Multi Level Learning. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 454–459. IEEE Computer Society, 2007.

[150] R. Rosipal and M. Gorilami. An adaptive support vector regression filter: A signal detection application. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN'99)*, volume 2, pages 603–607, Edinburgh, 1999.

[151] Y. Rotem, A. Wachs, and D. R. Lewin. Ethylene compressor monitoring using model-based PCA. *AIChE Journal*, 46(9):1825–1836, 2000.

[152] D. E. Rumelhart, G. E. Hintont, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[153] D. Ruta. *Classifier Diversity in Combined Pattern Recognition Systems*. PhD thesis, 2003.

[154] D. Ruta and B. Gabrys. An overview of classifier fusion methods. *Computing and Information Systems*, 7(1):1–10, 2000.

[155] D. Ruta and B. Gabrys. Neural Network Ensembles for Time Series Prediction. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 1204–1209, Orlando, USA, 2007. IEEE Computer Society.

[156] S. Schaal and C. G. Atkeson. Constructive Incremental Learning from Only Local Information. *Neural Computation*, 10(8):2047–2084, 1998.

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

[158] R. E. Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[159] J. Scheffer. Dealing with missing data. *Research Letters in the Information and Mathematical Sciences*, 3(1):153–160, 2002.

[160] J. H. Schmidhuber. Reinforcement learning with interacting continually running fully recurrent networks. In *Proceedings of the International Neural Network Conference (INNC)*, volume 2, pages 817–820, Paris, 1990.

[161] M. Scholz and R. Klinkenberg. Boosting classifiers for drifting concepts. *Intelligent Data Analysis*, 11(1):3–28, 2007.

[162] S. Serneels and T. Verdonck. Principal component analysis for data containing outliers and missing elements. *Computational Statistics and Data Analysis*, 52(3):1712–1727, 2008.

[163] H. Smith and P. Fingar. *Business Process Management: The Third Wave*. Meghan-Kiffer Press, Tampa, 1st edition, 2003.

[164] A. J. Smola and B. Schoelkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

[165] I. Stanimirova, M. Daszykowski, and B. Walczak. Dealing with missing values and outliers in principal component analysis. *Talanta*, 72(1):172–178, 2007.

[166] H. B. Su, L. T. Fan, and J. R. Schlup. Monitoring the process of curing of epoxy/graphite fiber composites with a recurrent neural network as a soft sensor. *Engineering Applications of Artificial Intelligence*, 11(2):293–306, 1998.

[167] J. A. K. Suykens. *Least Squares Support Vector Machines*. World Scientific, 2002.

[168] K. Syarov. *Adaptive multivariate statistische Methoden zur Prozessberwachung und -vorhersage*. Masters thesis, 2006.

[169] A. Tsymbal. The problem of concept drift: definitions and related work. Technical Report TCD-CS-2004-15, Departament of Computer Science Trinity College, Dublin, https://www.cs.tcd.ie/publications/techreports/reports, 2004.

[170] E. Tzanakou, R. Michalak, and E. Harth. The Alopex process: Visual receptive fields by response feedback. *Biological Cybernetics*, 35(3):161–174, 1979.

[171] N. Ueda and R. Nakano. Generalization error of ensemble estimators. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 1. IEEE, 1996.

[172] G. Valentini and F. Masulli. Ensembles of learning machines. In *13th Italian Workshop on Neural Nets*, volume 2486 of *Series Lecture Notes in Computer Sciences*, pages 3–22. Springer-Verlag, 2002.

[173] V. N. Vapnik. *Statistical learning theory*. Wiley, New York, 1998.

[174] V. Venkatasubramanian, R. Rengaswamy, and S. N. Kavuri. A review of process fault detection and diagnosis Part II: Qualitative models and search strategies. *Computers and Chemical Engineering*, 27(3):313–326, 2003.

[175] V. Venkatasubramanian, R. Rengaswamy, S. N. Kavuri, and K. Yin. A review of process fault detection and diagnosis Part III: Process history based methods. *Computers and Chemical Engineering*, 27(3):327–346, 2003.

[176] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri. A review of process fault detection and diagnosis Part I: Quantitative model-based methods. *Computers and Chemical Engineering*, 27(3):293–311, 2003.

[177] S. Vijayakumar, A. D'Souza, and S. Schaal. Incremental Online Learning in High Dimensions. *Neural Computation*, 17(12):2602–2634, 2005.

[178] R. Vilalta and Y. Drissi. A Perspective View and Survey of Meta-Learning. *Artificial Intelligence Review*, 18(2):77–95, 2002.

[179] R. Vilalta, C. Giraud-Carrier, and P. Brazdil. *Meta-learning*. Handbook of data mining and knowledge discovery in databases. Springer Verlag, 2005.

[180] P. Vorburger and A. Bernstein. Entropy-based Concept Shift Detection. In *Proceedings of the Sixth International Conference on Data Mining*, pages 1113–1118, 2006.

[181] B. Walczak and D. L. Massart. Robust principal components regression as a detection tool for outliers. *Chemometrics and Intelligent Laboratory Systems*, 27(1):41–54, 1995.

[182] B. Walczak and D. L. Massart. Dealing with missing data Part I. *Chemometrics and Intelligent Laboratory Systems*, 58(1):15–27, 2001.

[183] B. Walczak and D. L. Massart. Dealing with missing data: Part II. *Chemometrics and Intelligent Laboratory Systems*, 58(1):29–42, 2001.

[184] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 226–235, 2003.

[185] L. Wang, C. Shao, H. Wang, and H. Wu. Radial Basis Function Neural Networks-Based Modeling of the Membrane Separation Process: Hydrogen Recovery from Refinery Gases. *Journal of Natural Gas Chemistry*, 15(3):230–234, 2006.

[186] S. Wang and J. Cui. Sensor-fault detection, diagnosis and estimation for centrifugal chiller systems using principal-component analysis method. *Applied Energy*, 82(3):197–213, 2005.

[187] S. Wang and F. Xiao. AHU sensor fault diagnosis using principal component analysis method. *Energy and Buildings*, 36(2):147–160, 2004.

[188] X. Wang, U. Kruger, and G. W. Irwin. Process monitoring approach using fast moving window PCA. *Industrial and Engineering Chemistry Research*, 44(15):5691–5702, 2005.

[189] X. H. Wang, S. Y. Li, and Y. G. Xi. Soft sensor modeling for slab temperature estimation. In *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ'03)*, volume 2, 2003.

[190] Y. Wang and G. Rong. A self-organizing neural-network-based fuzzy system. In *Proceedings of Fifth International Conference on Artificial Neural Networks*, pages 106–110, 1997.

[191] K. Warne, G. Prasad, S. Rezvani, and L. Maguire. Statistical and computational intelligence techniques for inferential model development: a comparative evaluation and a novel proposition for fusion. *Engineering Applications of Artificial Intelligence*, 17(8):871–885, 2004.

[192] K. Warne, G. Prasad, N. H. Siddique, and L. P. Maguire. Development of a hybrid PCA-ANFIS measurement system for monitoring product quality in the coating industry. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, volume 4, 2004.

[193] S. M. Weiss and C. A. Kulikowski. *Computer Systems that Learn*. Morgan Kaufmann Publishers, 1991.

[194] G. Welch and G. Bishop. An Introduction to the Kalman Filter. Technical Report Technical Report TR 95-041, University of North Carolina- Department of Computer Science, 1995.

[195] P. J. Werbos. *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. PhD thesis, 1974.

[196] G. Widmer. Tracking Context Changes through Meta-Learning. *Machine Learning*, 27(3):259–286, 1997.

[197] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, 1996.

[198] S. Wold, M. Sjoestroem, and L. Eriksson. PLS-regression: a basic tool of chemometrics. *Chemometrics and Intelligent Laboratory Systems*, 58(2):109–130, 2001.

[199] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[200] W. Yan, H. Shao, and X. Wang. Soft sensing modeling based on support vector machine and Bayesian model selection. *Computers and Chemical Engineering*, 28(8):1489–1498, 2004.

[201] S. H. Yang, B. H. Chen, and X. Z. Wang. Neural network based fault diagnosis using unmeasurable inputs. *Engineering Applications of Artificial Intelligence*, 13(3):345–356, 2000.

[202] Y. Yang and T. Chai. Soft sensing based on artificial neural network. In *Proceedings of the 1997 American Control Conference*, volume 1, 1997.

[203] E. Zamprogna, M. Barolo, and D. E. Seborg. Development of a soft sensor for a batch distillation column using linear and nonlinear PLS regression techniques. *Control Engineering Practice*, 12:917–929, 2004.

[204] E. Zamprogna, M. Barolo, and D. E. Seborg. Estimating product composition profiles in batch distillation via partial least squares regression. *Control Engineering Practice*, 12(7):917–929, 2004.

[205] H. Zhang and B. Lennox. Integrated condition monitoring and control of fed-batch fermentation processes. *Journal of Process Control*, 14(1):41–50, 2004.

[206] L. Zhao and T. Chai. Adaptive Moving Window MPCA for Online Batch Monitoring. In *Proceedings of the fifth Asian Control Conference*, volume 2, 2004.

[207] Z. Zivkovic and F. van der Heijden. Recursive unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):651–656, 2004.