Unsupervised Pattern Recognition Methods for Exploratory Analysis of Industrial Process Data

Esa Alhoniemi

Helsinki University of Technology
Department of Computer Science and Engineering
Laboratory of Computer and Information Science
P. O. Box 5400
FIN-02015 HUT
Finland

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission for public examination and debate in Auditorium T2 at Helsinki University of Technology (Espoo, Finland) on the 13th of December, 2002, at 12 noon.

Alhoniemi, E., **Unsupervised pattern recognition methods for exploratory analysis of industrial process data**, doctoral thesis, Helsinki University of Technology, Espoo 2002, 65 pp. ISBN 951-22-6242-8. UDC 004.032.26:004.93:66.102:519.237.8

Keywords

self-organizing map, cluster analysis, exploratory data analysis, pattern recognition, process monitoring, industrial applications, unsupervised learning

Abstract

The rapid growth of data storage capacities of process automation systems provides new possibilities to analyze behavior of industrial processes. As existence of large volumes of measurement data is a rather new issue in the process industry, long tradition of using data analysis techniques in that field does not yet exist. In this thesis, unsupervised pattern recognition methods are shown to represent one potential and computationally efficient approach in exploratory analysis of such data.

This thesis consists of an introduction and six publications. The introduction contains a survey on process monitoring and data analysis methods, exposing the research which has been carried out in the fields so far. The introduction also points out the tasks in the process management framework where the methods considered in this thesis – self-organizing maps and cluster analysis – can be benefited.

The main contribution of this thesis consists of two parts. The first one is the use of the existing and development of novel SOM-based methods for process monitoring and exploratory data analysis purposes. The second contribution is a concept where cluster analysis is used to extract and identify operational states of a process from measured data. In both cases the methods have been applied in exploratory analysis of real data from processes in the wood processing industry.

Acknowledgments

Most of all, I sincerely thank Professor Olli Simula for his supervision, guidance, and support during my work. I am as well grateful for Academician Teuvo Kohonen and Academy Professor Erkki Oja for making the Laboratory of Computer and Information Science together with the Neural Networks Research Centre an excellent place to carry out scientific research. I also thank all the other people in the laboratory and in the research centre: the inspiring and pleasant atmosphere is something I will never forget. Especially, I wish to thank the current and the past members of the Intelligent Data Engineering research group, in particular Mr. Jussi Ahola, Mr. Johan Himberg, Mr. Sampsa Laine, and Mr. Jukka Parviainen.

The work of this thesis has been carried out in several projects funded by Tekes (the National Technology Agency) and industrial project partners. I would like to thank all the financiers as well as all the people in the companies who have participated in the projects. I thank Mr. Olli Ahava, Mr. Mats Backman, and Mr. Heikki Öhman who supported me in the project at UPM-Kymmene Wisaforest pulp mill. I am also grateful to Professor Risto Ritala, Mrs. Maija Federley, and Mr. Mika Suojärvi from KCL as well as Mr. Pekka Aho and Mr. Mikko Vuori from UPM-Kymmene Jämsänkoski paper mill.

The preliminary reading of this thesis was carried out by Professor Heikki Hyötyniemi, Helsinki University of Technology, and Professor Jussi Parkkinen, University of Joensuu. I am grateful for both reviewers for their valuable comments which led to considerable improvements of the thesis. Also, I owe my thanks to Professor Jaakko Hollmén, Dr. Samuel Kaski, and my roommate Dr. Juha Vesanto who all read early versions of my thesis and gave me invaluable comments and feedback. I also thank Mr. Jorma Kontio for helping me with the proofreading of my thesis in the final stage.

The additional financial support by the Finnish Cultural Foundation and Nokia Foundation is gratefully acknowledged.

Finally, I wish to thank my parents for their support and my dear wife Katri as well as our dear son Tuomas for always being there for me.

Turku, November 26, 2002

[illegible]

Esa Alhoniemi

Contents

List of publications

This thesis consists of an introduction and the following six publications.

1. Esa Alhoniemi, Jaakko Hollmén, Olli Simula, and Juha Vesanto (1999). Process Monitoring and Modeling Using the Self-Organizing Map. In *Integrated Computer-Aided Engineering*, Vol. 6, No. 1, pp. 3–14.

2. Juha Vesanto, Johan Himberg, Esa Alhoniemi, and Juha Parhankangas (1999). Self-organizing Map in Matlab: the SOM Toolbox. In *Proceedings of the Matlab DSP Conference 1999*, Espoo, Finland, November 16–17, pp. 35–40.

3. Esa Alhoniemi. Analysis of Pulping Data Using the Self-Organizing Map (2000). In *Tappi Journal*, Vol. 83, No. 7, p. 66. The paper is available in its entirety at TAPPI's web site at http://www.tappi.org/.

4. Juha Vesanto and Esa Alhoniemi. Clustering of the Self-Organizing Map (2000). In *IEEE Transactions on Neural Networks*, Vol. 11, No. 3, pp. 586–600.

5. Esa Alhoniemi and Olli Simula (2001). Interpretation and Comparison of Multidimensional Data Partitions. In *Proceedings of the 9th European Symposium on Artificial Neural Networks*, Bruges, Belgium, April 25–27, pp. 277–282.

6. Maija Federley, Esa Alhoniemi, Mika Laitila, Mika Suojärvi, and Risto Ritala (2002). State Management for Process Monitoring, Diagnostics and Optimization. In *Pulp & Paper Canada*, Vol. 103, No. 2, pp. 40–43.

This numbering is used in the main text when referring to the publications.

Abbreviations

BMU	Best-Matching Unit
DCS	Distributed Control System
FDI	Fault Detection and Isolation
HSV	Hue-Saturation-Value color system
K-S	Kolmogorov-Smirnov
PC	Principal Component
PCA	Principal Component Analysis
PLS	Partial Least Squares
SOM	Self-Organizing Map
SPC	Statistical Process Control
SSE	Sum of Squared Errors

Notations

$\alpha(t)$	learning rate function
ε	distinguishability constant, small positive number
$\sigma(t)$	neighborhood width function
Δ	step size in the bisection search algorithm
d	number of variables, i.e., data dimension
$d(\mathbf{x}, X_i)$	average distance between vector $\mathbf{x}$ and vectors in partition X_i
$d(\mathbf{x}_i, \mathbf{x}_j)$	distance between data vectors $\mathbf{x}_i$ and $\mathbf{x}_j$
c	index of the best-matching (or winner) unit
$h_{ci}(t)$	value of neighborhood function at unit i when the winner unit is c
i, i', j, k, l, m	indices used in several contexts
n_i	number of data vectors in ith partition or segment
s_i	tendency of ith segment to contain an outlier or transient
s_{lim}	threshold value for s_i
t	time index
t_0, t_f	start and end points of a segment
w	window size
C	number of clusters
$D_k(X_i, X_j)$	Kolmogorov-Smirnov statistic of kth variable between partitions X_i and X_j
D_k^{tot}	discrimination ability of kth variable
E_{lb}, E_{ub}	lower and upper bounds of error
E	error or distortion measure
$F_{k,i}(l)$	cumulative distribution function of kth variable in ith partition
K	number of segments
M	number of map units
N	number of data vectors
V_i	Voronoi set of map unit i
X_i	ith partition
$x_i(t)$	tth sample in ith time series (or signal)
$\mathbf{m}_i$	model vector of map unit i
$\mathbf{r}_i$	position vector of map unit i on the map grid
$\mathbf{x}$	arbitrary data vector
$\mathbf{x}_i$	ith data vector
$\mathbf{x}(t)$	tth sample in multivariate time series

Chapter 1

Introduction

1.1 Background and motivation

Data storage capacities of modern process automation systems have rapidly grown during the last decade. Nowadays, the systems are able to frequently carry out even thousands of measurements in parallel and to store them in databases. Because existence of large volumes of data is a rather new issue in the process industry, there does not exist long tradition of using computationally efficient data analysis techniques in that field. The work reported in this thesis is motivated by practical applications in the wood processing industry, where unsupervised pattern recognition methods were used to explore large records of process data. In order to understand the nature of this approach, it is appropriate to refer to the definition of unsupervised learning given by Kohonen in [59, p. 400].

> Learning without a priori knowledge about the classification of samples; learning without a teacher. Often the same as formation of clusters, whereafter these clusters can be labeled.

This thesis is about exploratory methods for finding and explaining structure of data describing behavior of complex industrial processes. This approach can be seen to have two major goals. Firstly, analysis of the structure of the data may reveal some completely new features of the process considered. Secondly, even though the behavior of the process would be well known, the unsupervised methods may still prove to be valuable tools to refine the information produced by numerous sensors of the process into a form that is easy to interpret for a human.

In Figure 1.1, a simplified representation of the management scheme of a computer-controlled process is shown. Data flow between different parts of the system is shown by arrows and the parts of the diagram which are covered in this thesis – process data, analysis of the data, and process monitoring – are emphasized by gray color. All parts of the system are briefly described below.

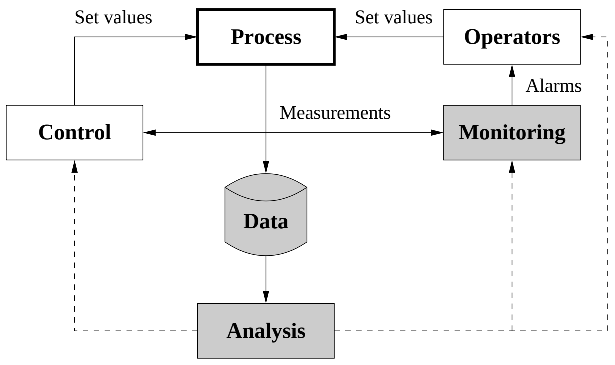


Figure 1.1: A simplified representation of the parts of a modern industrial process and data flows between them.

Data. Computer-controlled systems of today have large storages of measurement data, which are usually either time series or images. In this thesis, only time series data are considered: the term signal is in many contexts used as a synonym for such data.

Monitoring. Process monitoring is mainly responsible for process fault diagnosis. In the simplest case it is manually carried out by the operators using trend displays of measurement signals. However, if the number of the signals is large, computer-based tools can be used to support the operators to better understand the status of the process.

Analysis of process data. Analysis of process data means exploitation of process measurement databases to obtain new knowledge of the process or to summarize the properties of the data. The extracted knowledge can be utilized in adjustment of the process controls, the monitoring system, and training of the operators. The analysis can be seen as off-line monitoring the goal of which is to continuously improve the performance of the process in the long run by exploiting the process database. In the literature, sometimes also term "data mining" [8, 110] is used in this context. This thesis mainly focuses on exploratory data analysis which can be seen as an initial phase of data mining that aims at understanding and generating hypotheses of the data.

Control system and operators. A modern complex industrial process is controlled by both a computer-based distributed control system (DCS) [68, pp. 365–378] and human operators. The control system consists of several control loops, which use measurements of the process to automatically determine appropriate control signals for the actuators in the loops. In steady-state operation of the process, the DCS is usually responsible for most control routines, leaving only the decisions on the highest level for the human operators. In unexceptional situations like process startup, shutdown, or fault, the operator may also need to participate on the low-level operations.

1.2 Contributions of this thesis

1.2.1 Publications

In **Publication 1**, different ways to utilize the Self-Organizing Map (SOM) in analysis, monitoring, and modeling of complex industrial processes are considered. The methods are illustrated using three case studies. The author was responsible for the process data analysis experiments using data from a pulp mill and to a large degree wrote the general parts of the paper with Dr. Juha Vesanto.

In **Publication 2**, a freely available software package, the SOM Toolbox, is introduced. The author was responsible for implementation of all the input-output functions and partially implemented some utility functions for data structures and initialization routines of the SOM.

Publication 3 reports results of a case study where faults of a continuous pulp digester are analyzed using the SOM. The author was responsible for data acquisition, experiments, and writing the article. Aid for the interpretation of the results was provided by personnel of the pulp mill and Pulp Center, research institute located at the mill.

In **Publication 4**, direct clustering of data using classic hierarchical and partitive algorithms is compared with clustering of the SOM trained with the same data. The obtained experimental results suggest that using the SOM as an intermediate step in the clustering does not remarkably affect the clustering result but significantly improves scalability of the hierarchical clustering algorithms. The paper was joint work with Dr. Vesanto; the contribution of Dr. Vesanto was slightly greater due to novel pruning algorithm for the hierarchical clustering proposed in the paper.

Publication 5 describes a method for interpretation (e.g., characterization) and comparison of multidimensional data partitions. For example, the partitionings obtained by clustering algorithms can be analyzed using the proposed methods. The author was responsible for the original idea and the experiments. The paper was mainly written by the author and edited by the coauthor prof. Olli Simula.

In **Publication 6**, cluster analysis is used to extract and identify process states from measured data. The author wrote the parts of the article which describe the cluster analysis of the process data and was responsible for the implementation of the methods used in the experiments. The clustering algorithms can be found from the literature as such, but the author developed and implemented some novel methods for data preprocessing and interpretation of the clusters. The so far unpublished preprocessing methods are now described in Section 4.2 of this thesis. The most important interpretation methods are described in Publication 5.

1.2.2 Other contributions

As stated above, the introductory part of this thesis also contains previously unpublished material. The most central such a thing is segmentation and selection of segments before clustering of process data which is described in Section 4.2. Also, the cluster analysis procedure presented in Publication 6 is considered in more depth in Section 4.2.

In addition, the concept introduced in Section 3.4 where the SOM is trained using aggregate features of process signals – which makes it possible to summarize properties of all signals of a process database – has not so far been published elsewhere.

1.3 Structure and organization of this thesis

The rest of this thesis is organized according to the illustration shown in Figure 1.2. Chapter 2 contains a literature review in the field of process monitoring and analysis of process data. The chapter brings out the problems where the methodology considered in this thesis can be used, and discusses how the obtained results can be deployed. In Chapter 3, management of process data and properties of the data are considered: in order to preprocess the data properly, it is essential to understand the nature of the process data. The unsupervised methods are described in Chapter 4. These include the SOM (Section 4.1) and cluster analysis (Section 4.2). The SOMs are trained using raw process measurements but the cluster analysis is preceded by a feature extraction phase which is also described in Section 4.2. Finally, Chapter 5 contains conclusions and discussion on possible directions of future research.

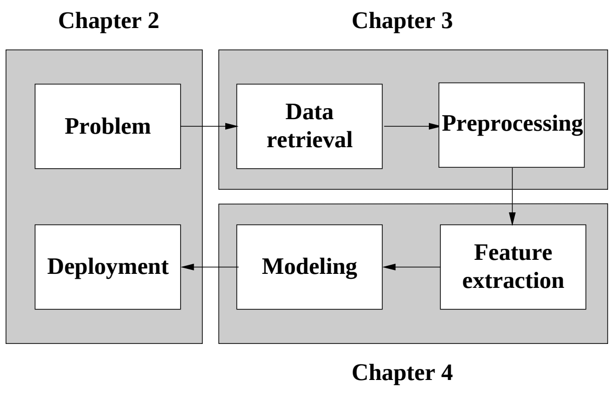


Figure 1.2: The phases of modeling of process data.

Chapter 2

Process management

Both process management tasks considered in this thesis deal with observation of the behavior of a process. In [74, pp. 23–25], the two approaches are called by a common name process monitoring, but are here distinguished as follows:

- *monitoring* is an on-line task which aims at indication of faults and other anomalies in a process, and
- *analysis of process data* means improvement of a process in the long term based on off-line analysis of historical process data.

In Table 2.1, characteristics of the two tasks are summarized. Process monitoring methods have been extensively studied during the past few decades whereas analysis of process data – sometimes also called process data mining [110] – is a rather new and emerging application field. The reason for this is simple: collection of large volumes of measurement data has not been possible for more than about a decade due to lack of efficient data mass storages.

	Monitoring	**Analysis of process data**
Type	On-line	Off-line
Time span	From seconds to hours	From a day to a year
Uses	Recent measurements	Historical process data
Carried out by	Operation personnel	Process engineer or analyst
Performed	Continuously	Infrequently
Primary goal	Fault diagnosis	Process improvement

Table 2.1: Characteristic features of the process management tasks considered in this thesis.

2.1 Process monitoring

The following definition of process monitoring was given in [46], where an attempt to clarify definitions of some central terms in the field was made.

> Monitoring is a continuous real-time task of determining the conditions of a physical system, by recording information, recognizing and indicating anomalies in the behavior.

In other words, the purpose of the monitoring is to indicate whether a process has deviated from its acceptable state, and if it has, why. The deviations are called process *faults*. Observation of the faults is known as *fault detection*, which is followed by *fault isolation*, determination of the location and the type of the fault [46]. Fault Detection and Isolation (FDI) – also known by a common name *fault diagnosis* [28, p. 6] – can be carried out in many ways. The three logical parts of any FDI scheme shown in Figure 2.1, namely detection, decision and isolation, may be partially integrated. Fault detection takes as input the current values of the process measurements and produces one or more fault indicator signals, which are often called *residuals*. After the detection phase there is an inference mechanism which takes the fault indicator(s) as input and decides whether a fault has occurred or not. Detection of a fault is followed by an isolation phase which carries out identification of the fault. Comprehensive reviews and surveys on process monitoring can be found in [45, 46, 84, 97]. Additionally, many books on the subject exist, see for example [15, 38].

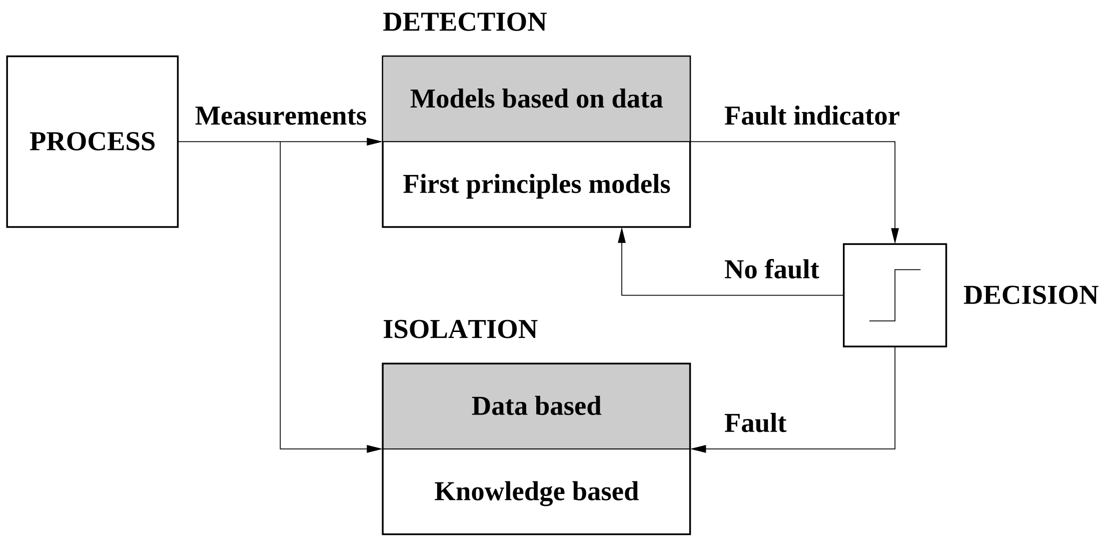


Figure 2.1: The three phases of the FDI: detection, decision, and isolation. The model families which include the methods considered in this thesis have been emphasized using gray color.

2.1.1 Fault detection

Fault detection methods are in this thesis divided into two categories: first principles process models (see for example [70]) and models of process data. In the former approach, physical structure and *a priori* known relationships between variables of a process form the basis for the construction of the model and observed data are not required. In the latter case, the structure of the model is generic or depends on the data and the model is based on observed data produced by the sensors of the process. However, in practice the line between these two categories is not sharp: measurement data may be used in construction of a first principles model and, correspondingly, *a priori* knowledge of a process can be used in the construction of a model using process data.

Use of data-driven models instead of the first principles models is justified if construction of an accurate first principles model is (1) impossible due to unknown process phenomena or (2) computationally infeasible because of complexity of the process. In addition, if the modeling using first principles would be possible but laborious, use of a data-based model may still be reasonable choice if the process is modified often: a model based on data is typically easier to update than a first principles model.

First principles models

First principles models are beyond the scope of this thesis. However, as they are commonly used in fault detection, they deserve to be briefly mentioned. Excellent surveys on the use of such models in fault diagnostics can be found in [24, 44]. The first principles models are often divided into the following three categories.

- *State/output observers or estimators* estimate the system states or outputs based on known system inputs and outputs or states. The residual is the difference between the estimated and the measured state or output.
- *Parity relations* can be used to check the consistency of the model with the measurements. For example, value of a process output measurement can be estimated using measured input and a process model. The residual is the difference between the measured and the model output.
- In *parameter estimation*, the residuals are differences between known model parameters and the same parameters estimated from measured data.

Models of process data

A common factor for all the models of process data is that they are generic in their structure and model the process in a data-driven manner, not using physical properties of the process. Here, the data-based models have been divided into two categories, static and dynamic models, which are both described below. Of these, the latter one is beyond the scope of this thesis.

Static models. The static models can be further divided into unsupervised and supervised models. The former ones aim at finding a faithful presentation of the normal operation space of the process by treating all variables in the same way whereas the latter ones are typically regression models between input and output measurements of a process. Because this thesis is focused on the unsupervised models, they are mainly considered here.

In essence, unsupervised modeling means computing a model of the process data manifold. The fault indicator is a measure of similarity between the current measurements of the process and the manifold. As shown in Section 4.1, the SOM can be used for that purpose, making it possible not only to model but also to visualize the high-dimensional data by projecting it in two dimensions. However, if one is exclusively interested in measuring the similarity between the current measurements and the data manifold, some pure density estimation method might as well be used.

In the process industry, so-called Statistical Process Control (SPC) methods are widely used. All the variations within a process are either seen as random (common cause) or non-random (special cause) variations [82, p. 68]. Parameters of an SPC model are computed using data describing a period when no special cause variations in the process exist. The obtained model can then be used to detect unexpected special cause variations of the process.

- Univariate SPC methods are usually used to monitor quality of process output by detecting changes in signal properties, typically mean or variance [82, p. 105], but they are impractical for monitoring a multivariate complex process, because each measurement requires a model of its own and dependencies between variables are ignored.

- The multivariate SPC methods are basically linear dimension reduction techniques. The most commonly used SPC method is the Principal Component Analysis (PCA), which is based on eigenvalues and -vectors of the covariance matrix of the data, see [49]. The PCA can be used to compute optimal dimension reduction in the sense of loss of variance by projecting the data onto the subspace spanned the eigenvectors with greatest eigenvalues. The obtained model can then be used in fault detection using two statistics. The so-called Q statistic is a measure of changes in the correlation structure of the data and the Hotelling's T^2 depicts deviation of the process from the distribution mean provided that the correlation structure has remained unchanged.

  The Partial Least Squares (PLS) algorithm is a supervised algorithm, but it is mentioned here, because it is widely used and very similar to the PCA. The difference is that it carries out dimension reduction which maximizes the covariance between input and output data. For a review on use of the PCA and PLS in process monitoring, see for example [71]. Book by Chiang *et al.* also contains a review on many variants of both the PCA and the PLS [15].

Dynamic models. The dynamic models include signal and system models, which differ from the static models in the sense that the models have memory, i.e., the output of the model never depends on the current inputs only. The goal of a signal model may be, for instance, to estimate the "true value" of the signal which is corrupted by noise. A typical system model is a black-box model which is computed using available input and output data of the modeled system, that is, using system identification methodology, see for example [69, 81]. In the former case, the estimated signal value itself can be a fault indicator whereas in the latter case, a sensible choice is the difference between the measured system output and the output of the model obtained using the measured inputs.

2.1.2 Decision

The decision phase determines whether there is a significant change in the fault indicator signal. The decision method is often tightly coupled with the fault detection method. In the case of the first principles models, it may be simply a threshold for the residuals [84]. The decision can also be, for instance, based on a change in the mean or the variance of some measurement signal – or change in a parameter of an adaptive model that is estimated based on measured data. The change detection issues are considered in [9, 28].

2.1.3 Fault isolation

Fault isolation can be carried out in many ways. However, the knowledge for the isolation can be obtained in two principally different ways: (1) using external knowledge like documentation of the process, experiences of the operating personnel, or principles of chemical engineering science, or (2) based on process data [97]. Approach (1) leads to, for example, traditional expert systems where all the knowledge needs to be transformed into form of if–then rules in the knowledge base. The fundamental principle of approach (2) is to learn knowledge from examples, i.e., observed data. The first approach, the traditional expert systems, are beyond the scope of this thesis and are not discussed here. Within the data-based methods, the following two main streams can be distinguished.

Classification. If labeled samples are available, it is possible to use these data to construct a classifier in which each class represents one fault type. Once a new sample consisting of the current measurements of the process is given to the classifier, it gives the class, that is, the type of the fault as an output. If the classifier is complex, for example a multilayer perceptron network [96], it is a black-box model with an inference mechanism that is hard to interpret and understand for a human. References to many classification algorithms can be found in the pattern recognition literature, see for example [12, 20, 89].

If the class labels of the data vectors are not available, the unsupervised methods considered later in Chapter 4 can be used to model the structure of the data. Using the model all the data vectors can be given an interpretation, i.e., labeled, and used as a basis of a classifier.

Rules from data. In traditional expert systems the rules are defined based on external information on the process, but it is also possible to learn rules from data. This approach is opposite to the classification approach mentioned above because the rules can be readily understood by the humans. For example, the SOM can be used for that purpose, see [86, 94]; for other possible methods, see e.g. [110, pp. 173–191]. If labeled data are available, it is possible to construct a decision tree which is actually a tree-structured classifier with a splitting rule in each node [20, 89].

2.1.4 SOM in process monitoring

In this section, process monitoring applications of the SOM are reviewed. The SOM is considered in detail in Section 4.1, but at this point it is sufficient to know that the SOM roughly models the input data manifold and makes it simultaneously possible to display the data in two dimensions for visualization purposes. Process monitoring is carried out by projecting the current measurement value on that display: location of the projection on that display indicates the state of the process.

Different possibilities to use the SOM in process monitoring are illustrated in Figure 2.2. The four approaches are discussed in more detail below with references to works where the corresponding methodology has been used.

Approach (a) One map is trained using all possible data of the monitored process. The position of the projection of the current measurement vector on the map directly indicates the presence of the fault and its type [54, 78, 80]. The main drawback of this approach is that samples from the whole operational space with all possible faults are usually difficult to obtain.

Approach (b) There is one map of the normal operation space of the process for fault detection [2, 31, 43, 54, 103, 112], or multiple maps, if it is known that the process has more than one known normal operation mode [32]. If desired, fault detection can be carried out by a separate map or maps for isolation of the fault [104].

Approach (c) This approach, unlike the two previous ones, is not part of the FDI scheme. Here the SOM is trained using the data depicting normal operation of the process and can be used either (1) to indicate the operation point of the process to human operator/analyst [77, 88, 101] or (2) to use the indication of the operation point as a part of a more complex system [67].

Approach (d) As shown in Publication 1, the SOM can be used as a so-called soft sensor to estimate missing values of a data vector. In the process monitoring context, the available data are on-line measurements of a process and missing values are either values of off-line laboratory analyses [65] or classes [26]. The basic idea is that the SOM is trained using data where all the values are present. As the on- and off-line measurements are interrelated, the SOM learns the association between them whereafter it can be used to predict the off-line values (or to classify) given only the on-line measurements. This construction is called the "supervised SOM" [59].

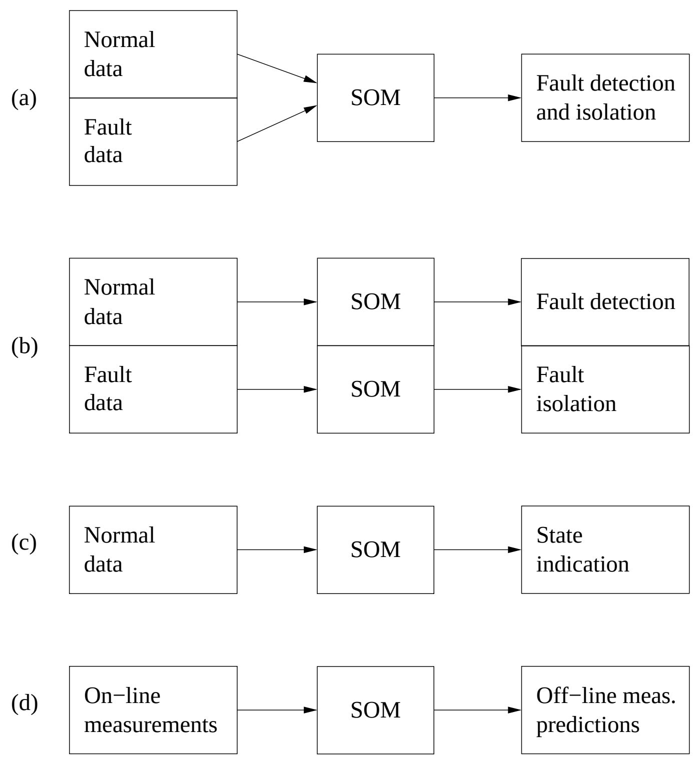


Figure 2.2: Four possibilities to use the SOM in process monitoring.

2.2 Analysis of process data

Goal of the analysis of process data is to extract useful knowledge from the data. This definition is closely related to the one of data mining given in [30, p. 1].

> Data mining is the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner.

In the process management framework, the obtained knowledge can be used to, for example, enhance automatic process controls, improve the knowledge base of the monitoring system, or gain experience of the process operators and engineers. Usually, the data describe regular operation of the process, and they don't have to – even though they may – be generated using specific process experiments.

It should be acknowledged that the primary goal of the methods considered in this thesis is exploratory analysis of the data, that is, summarization and visualization of the properties of the data for a human analyst. The analyst can then better understand the data, and possibly use the obtained information further to generate hypotheses, which can be verified using either *a priori* knowledge of the process or other methods.

The methods of this thesis can be used for two purposes: either to discover dependencies between variables, or to find groups of similar data vectors. As shown later in Section 4.1, the SOM can be efficiently used for both tasks whereas the cluster analysis is a methodology for detecting groups of similar data vectors only. The tasks in the process management framework in which the methods considered in this thesis can be used are described below.

2.2.1 Grouping of data

Analysis of operator actions. Even though control procedures of a modern process are mainly automated, the operators usually have manual control over many central process variables. It is possible that a product with acceptable quality can be produced using several different combinations of these controls. Cluster analysis can be used to discover the different combinations, making it possible for the operators to learn to use the most profitable control setting in each situation. This was one of the motivations of the study reported in Publication 6 where cluster analysis was applied for the task. The obtained clusters were analyzed using the method presented in Publication 5.

Other applications. Clustering of data using the SOM is considered in Publication 4. This methodology has been used in the analysis of mobile network data [87], but it seems that applications in the process industry have not been reported in the literature so far. However, the ability of the SOM to group similar data has been applied in process control related issues [48, 102] and grouping of process faults [25, 41, 48].

Cluster analysis methodology has been used in, for example, adaptive compensation of seasonal variations of a process in monitoring [100], and extraction of previously unknown operational states of processes [99, 111]. In some closely related application fields both the SOM (for example in chemistry [113]) and cluster analysis (for example in fault diagnosis [15]) are widely used methods.

2.2.2 Dependencies of variables

Analysis of faults of a continuous pulp digester. The SOM can efficiently be used for visualization of dependencies and interrelations between variables of high-dimensional data in various ways, which are considered in detail later in Section 4.1. Methods of this type are presented in Publications 1 and 3. The latter one contains results of a study where the visualizations were used in analysis of the dependencies between an output variable and state measurements of a pulping process in exceptional process conditions.

Other applications. Very few industrial applications of this type have been earlier reported in the literature. So far the SOM has been used in analysis of production processes of very large-scale integration circuits [73], visualization of fluid-bed granulation process [88], and process faults [41].

Chapter 3

Process data

3.1 Data storage and retrieval

In modern automation systems, most physical quantities are automatically and continuously measured and stored in databases. Even though the set values of actuators do not measure any physical phenomenon of the process, they are sometimes stored, too. Laboratory measurements are exceptional in the sense that they are usually performed irregularly and fed into databases manually. The last and most indefinite class of data consists of categorical quantities which describe status of the process or some part of it, for example, the identifier of the paper grade which is currently been produced. It is worth noticing that the stored data may be averaged, possibly in several different time resolutions, in order to save storage space. In Table 3.1, a categorization of different types of measurements is presented. One typical signal from each category is shown in Figure 3.1.

Data type	**Values obtained**	**Data source**	**Noisy**	**Spacing in time**	**Scale**
Physical quantity (continuous)	automatically	process phenomenon	yes	regular	interval
Physical quantity (laboratory)	manually	process phenomenon	yes	irregular	interval
Set value	automatically	control system or operator	no	regular	interval
State indicator	automatically	control system or operator	no	regular	nominal

Table 3.1: Summary of properties of variables in a process database.

A large industrial process is in practice not a monolithic system but merely a set of strongly interconnected processes. It is not rare that these processes are run by different control systems which all may have measurement data storages of their own. Data are

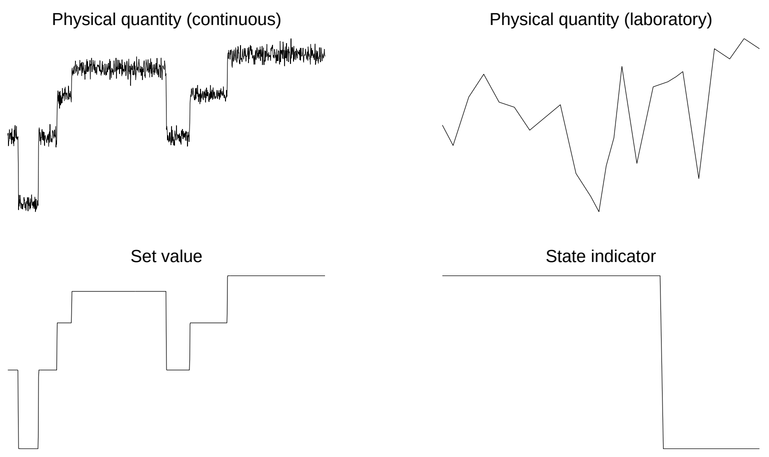


Figure 3.1: Different types of signals obtained from a database of a pulp mill: raw material feed (top left signal), corresponding set value (bottom left), a quality measure of pulp which is measured in laboratory (top right) and a signal indicating on/off status of an automatic control (bottom right). Time axis is the same in all four signals.

retrieved from these databases using queries where the time labels of the first and the last sample, the variables to be retrieved, and the resolution are specified. If data are obtained from many sources, there are naturally several separate data sets after retrieval of the data. Fortunately, each record actually consists of two fields, a time label and the corresponding value. Using the time labels it is quite straightforward to adjust the data to finally form only one data matrix, which contains a multidimensional time series.[1] The rows of the matrix are called *data (*or *measurement) vectors*. Correspondingly, columns of the matrix are one-dimensional time series, that is, measurement values of individual sensors at different time instances.

3.2 Properties of process data

This section brings out some properties of the process data which are worth noticing in any task where the process data are used. In the pattern recognition framework *a priori* knowledge on the deficiencies of the data is especially useful in the preprocessing phase, which aims at maximizing the usefulness and the quality of the data.

Constant values. If the value of a measurement remains unaltered over some time period from one sample to another, it may be an indication of that

- there is a malfunction or shutdown of the process,

[1]An effi cient data analysis tool like, for example, KCL Wedge [56] makes use of several sources of data transparent by internally combining the data from different sources, which saves a large amount of otherwise required laborious manual work of the analyst.

- the sensor lies in a part of the process that is not used after modification of the process,
- measurements of the sensor are not available, but the data storage system is not able to register them as missing; in this case, often the last valid measurement or zero is stored, or
- the value of the measurement either over- or undershoots the scale of the sensor; if this is the case, the sensor gives greatest (or smallest) possible value of its scale as an output.

Dead time. Change in process input may affect output of the process after a delay which is known as *dead time*. As these delays are often at least approximately known, it is sensible to adjust the data in such a way that the corresponding input and output values are in the same data vector. For example, in the pulping process analyzed in Publication 3, the delays are considerably long and such adjustments are required.

Dynamics. The methods considered in this thesis are static, which means that they are suitable for analysis of data recorded during steady state operation of a process. It is important to acknowledge that the measurement data may as well originate from a process with slow dynamics. In that case, use of a static model may lead to misleading conclusions on the dependency between the measured input and output data if the state variables are not observed. For example, the pulp digester considered in Publication 3 has very slow dynamics. However, because only measurements describing the state and the output of the process were considered in the analysis, the dynamics did not constitute a problem.

Missing data. All the measurement values of a variable may not be available in the database. This may be due to

- a problem in data transfer between the sensor and the database,
- malfunction or shutdown of the data storage system,
- malfunction of the sensor,
- removal of the sensor from the process, or
- retrieval of measurement data of the sensor from the time before installation of the sensor.

The two first items in the list above typically produce short periods of missing values whereas in the other cases the period is usually considerably long.

Noise. The physical measurements are corrupted by noise. In [83, pp. 198–234], these errors are divided into internal and external errors according to their nature. The former are generated by random electric phenomena in the measurement devices. The latter are of human origin and can be totally eliminated or substantially reduced by care in design and engineering. An often made standard assumption is that the noise is additive and its distribution is Gaussian; such noise can be reduced by averaging.

Outliers. Outliers are observations with a unique combination of characteristics identifiable as distinctly different from the other observations. They may be classified into one of the following classes: (1) data entry error, (2) extraordinary observation with an explanation, (3) extraordinary observation without explanation, and (4) unique combination of values across the variables with all the variable values in the ordinary range [29, pp. 64–65].

Scales. Measured variables are quantities like temperatures, pressures, and flows which are typically presented in different scales. Because in this thesis the Euclidean distance is used as a measure of similarity between data vectors, it is necessary to make the variables commensurable. As the relative importance of the variables is usually unknown, a commonly used heuristical standardization, where mean of each variable is moved to zero and variance is scaled to unity is used. This gives each variable roughly equal weight in the computation of the distances between the data vectors.

Systematic errors. Systematic error is caused by one or more factors that systematically affect measurement of a variable. For example, some measurements tend to drift and require frequent calibration. In some cases, data reconciliation methodology can be used to optimally adjust measured data so that the adjusted values obey the conservation laws and other constraints [17, 90]. This naturally requires that redundant measurements exist for measured quantities and they are exactly known. If this is not the case, the systematic errors are very difficult to eliminate and they usually introduce error in any data-driven approach.

3.3 The data used in this thesis

In this thesis, real-world process data have been used in Publications 1, 3, and 6, in Section 3.4, and in Chapter 4. The data were in all cases time series obtained from the automation system of a real process, usually as ten minute averages. The data consisted of on-line measurements only, because available off-line laboratory measurements were carried out too sparsely, in many cases only once in a shift (a period of eight hours) or even less. The types of the on-line measurements used were temperatures, pressures, flows, rotation speeds, tank levels, and on-line quality measures.

Before any analysis method can be used, often some strongly application dependent preprocessing is necessary, because the data contain distortions and errors which would weaken the results of the analysis. Because the mechanisms which produce them are at least partially known beforehand, it may be possible to eliminate some of them during preprocessing of the data. In this thesis, for example, adjustment of pulping data due to long delays of the process was carried out.

If necessary, the preprocessing can be followed by a feature extraction phase, which transforms the data into a form that better describes the data from the problem point of view. This thesis deals with time series data, which can be analyzed either in the time or in the frequency domain. Both the self-organizing map and the cluster analysis methodologies considered in the next chapter were used in time domain, because it is a sensible approach for the problems at hand. However, these is no reason why the methods could not have been used with features based on frequency contents of the data – like spectral or wavelet features – if it was necessary.

Another important aspect of the feature extraction is the possibility to model dynamic phenomena using static methods like the SOM or the clustering algorithms. An often used approach is to slide a narrow window over time series or to segment the series into small time windows and then to compute descriptive features for each window. In Section 4.2, a segmentation approach was used to remove uninteresting signal parts.

3.4 Exploration of signals of a database

In this section, some aggregate features which can be used to characterize a time series with thousands of samples are considered. Using these features it is possible to obtain a rough overall view of hundreds of time series stored in a real process database. The view concretely shows that a real database is heterogeneous and includes measurements which may have potentially erroneous behavior of different kind. All the features are not – and in this case, they need not be – time invariant, because the idea is to display contents of the entire database and the same display is not used for new data.

Eleven statistics which are based on the properties of the process data described in the previous section are listed in Table 3.2. When the statistics of a time series $x(t)$ are combined into features it is possible to get a simplified representation of the properties of the time series. The features, which reflect potentially erroneous behavior of a measurement, are listed in Table 3.3.

The information obtained using these features is useful in maintenance of the process: by computing the features for all measurements, it is possible to efficiently find possibly broken sensors without sensible output. It would be obviously reasonable to verify which of the sensors are actually broken and to either fix or remove them, because they unnecessarily increase load of the data storage system.

Statistic	**Description**
t_0	index of the first non-missing value
t_f	index of the last non-missing value
$t_{\text{lastchange}}$	index of the last change in $x(t)$, i.e. $x(t_{\text{lastchange}} - 1) \neq x(t_{\text{lastchange}})$
N	total number of values
$N_{\text{available}}$	number of available (non-missing) values in $x(t)$
N_{missing}	number of missing values in sequence $x(t_0), \ldots, x(t_f)$
N_{max}	number of maximum values in $x(t)$
N_{min}	number of minimum values in $x(t)$
N_{unique}	number of unique values in $x(t)$ (excluding minima and maxima)
N_{mode}	number of occurrences of a value (excluding minima and maxima) which is most frequently present in $x(t)$
N_{changes}	number of changes in $x(t)$

Table 3.2: Variable statistics.

Feature	**Description**
t_0	installation of the sensor
t_f	removal or malfunction of the sensor
$\frac{t_{\text{lastchange}}}{t_f}$	removal or malfunction of the sensor
$\frac{N_{\text{missing}}}{t_f - t_0 + 1}$	reliability of the sensor
$\frac{N_{\text{max}} + N_{\text{min}}}{N_{\text{available}}}$	sensitivity to exceed the scale of the sensor
$\frac{N_{\text{unique}}}{N_{\text{available}} - N_{\text{max}} - N_{\text{min}}}$	behavior of the measurement
$\frac{N_{\text{mode}}}{N_{\text{available}} - N_{\text{min}} - N_{\text{max}}}$	behavior of the measurement
$\frac{N_{\text{changes}}}{N_{\text{available}}}$	behavior of the measurement

Table 3.3: Features based on the statistics in Table 3.2.

Database of a paper machine. In order to demonstrate use of the features in Table 3.3, database of a real full-scale paper machine was considered. The data consisted of 602 time series, each measured by a sensor of its own, with 107029 values each. As every measurement value is a ten-minute average, this means approximately a period of two years.

For each of the 602 signals, all the features presented in Table 3.3 were computed. Unfortunately, it is not easy to simultaneously display all the features in a readily understandable form. Therefore, even though the SOM is not considered in detail until in Section 4.1, it was used to produce a representation of the signals. However, to understand the illustration in Figure 3.2, one only needs to acknowledge that the SOM algorithm produces a regular two-dimensional "map" of data where similar signals – in the sense of used features – are arranged close to each other. In Figure 3.2, the map has been divided into ten regions or groups which represent signals with different features. For each group, one typical signal is shown.

Based on the illustration in the Figure 3.2, the sensors which have produced the data in the following groups might be sensible to check.

- The measurement values of the sensors which belong to the two groups in the middle of the map have not been available for a long time.

- The three groups in the top left corner of the map either get minimum or maximum values very often, which may be a consequence of frequent under- or overshooting of the scale of the measurement.

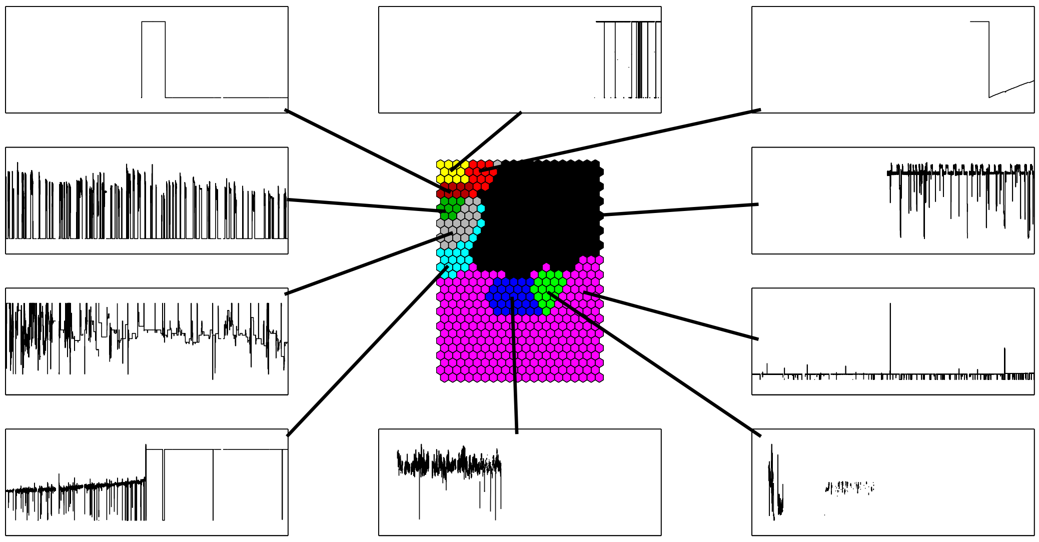


Figure 3.2: A SOM trained using the features of 602 signals. The SOM has been divided into ten regions which have been indicated using different colors. One example signal from each region is shown.

Chapter 4

Unsupervised methods

4.1 Self-Organizing Map

The Self-Organizing Map (SOM) [59] – also known as *self-organizing feature map* or *Kohonen map* – is an unsupervised learning neural network which carries out a topology preserving mapping from high-dimensional input data space onto a low-dimensional output grid. It has been widely used in engineering applications (for references, see [51, 59, 63, 93]), but also in various other problem domains like, for example, organization of text documents [62] and images [64].

In Publication 2, a freely available software package, the SOM Toolbox is briefly described.[1] The package contains the implementations of the training algorithms of the SOM and also a large set of different routines for visualization and interpretation of the maps. SOM Toolbox can also be used together with an earlier software implementation of the SOM, the SOM_PAK [61]. The illustrations which are later shown in this section are mainly produced using the SOM Toolbox.

4.1.1 Computation of the SOM

A SOM consists of M units (or neurons) which are arranged in a regular low-dimensional grid. The grid is typically two-dimensional because it is easy to visualize. Adjacent units on the grid are called *neighbors*. Each unit i is represented by a model (or prototype) vector $\mathbf{m}_i$ with dimensionality equal to that of input data and its position on the low-dimensional grid is denoted by $\mathbf{r}_i$.

During computation of the SOM, the "elastic net" – which consists of the map units – folds into the input data manifold. The model vectors become positioned in such a way that they roughly follow the density of the input data [59, pp. 152–159]. Hence, the SOM actually carries out two things. On one hand, it *quantizes* the input data using the model vectors, but on the other hand it also performs a *nonlinear projection* from the input data space to the units on the low-dimensional map grid. This grid can

[1] A more detailed description of the software can be found in [109].

be then used as a basis of many visualizations of the high-dimensional input data as illustrated later in this section.

Sequential algorithm

In the original, incremental version of the SOM algorithm [59, pp. 109–115], the input data vectors are presented to the algorithm one at a time in random order, possibly several times. For each input vector $\mathbf{x}$, the *best-matching unit* (BMU), denoted here by c, is determined:

$$c = \arg\min_{1 \le i \le M} \|\mathbf{x} - \mathbf{m}_i\|^2. \tag{4.1}$$

Usually, Euclidean distance is used as a measure of similarity $\|\cdot\|$. The model vector of the BMU, denoted here by $\mathbf{m}_c$, and its neighbors are moved toward the data vector in the original high-dimensional space according to formula

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \alpha(t) h_{ci}(t)[\mathbf{x} - \mathbf{m}_i(t)], \tag{4.2}$$

where the learning rate $\alpha(t) \in [0,1]$ is a decreasing function of time t and the neighborhood function $h_{ci}(t)$ is a non-increasing function around the BMU on the low-dimensional grid. Often, a Gaussian centered on the winner unit is used:

$$h_{ci}(t) = \exp\left(-\frac{\|\mathbf{r}_i - \mathbf{r}_c\|^2}{2\sigma(t)^2}\right). \tag{4.3}$$

Here $\mathbf{r}_c$ contains the coordinates of the winner unit c and $\mathbf{r}_i$ denotes coordinates of unit i on the low-dimensional map grid. During learning, both the learning rate and the width of the neighborhood $\sigma(t)$ are monotonically decreased.

Batch algorithm

Batch version of the SOM algorithm is computationally more efficient [60]. At each step of the algorithm, all the input data vectors are simultaneously used to update all the model vectors:

$$\mathbf{m}_i(t) = \frac{\sum_{j=1}^{M} n_j h_{ij}(t) \bar{\mathbf{x}}_j}{\sum_{j=1}^{M} n_j h_{ij}(t)}, \tag{4.4}$$

where $\bar{\mathbf{x}}_j$ is the mean of the data vectors in V_j, the Voronoi set of unit j, and n_j is the number of data vectors in V_j. Notation $h_{ij}(t)$ denotes the value of the neighborhood function at unit j when the neighborhood function is centered on the unit i.

In the batch algorithm, the learning rate function $\alpha(t)$ used in the sequential algorithm (Equation 4.2) is no longer needed, but the width of the neighborhood is monotonically decreased during the learning like in the sequential algorithm.

Distortion measure

Many algorithms aim at minimization of some cost or energy function. For the basic SOM algorithm it has been shown that such a function does not exist [22]. However, in the case of finite data and fixed neighborhood kernel, a distortion measure given by

$$E = \sum_{j=1}^{N} \sum_{i=1}^{M} h_{ci} \|\mathbf{x}_j - \mathbf{m}_i\|^2 \tag{4.5}$$

can be shown to be a local cost function of the SOM [58].

The update step of the sequential SOM algorithm (Equation 4.2) corresponds to a gradient descent step in minimization of Equation 4.5 provided that the winner unit $c = c(\mathbf{x}_j)$ of every data vector $\mathbf{x}_j$ remains unaltered. However, when the BMU of any of the data vectors changes, the cost function changes slightly and the SOM algorithm gives only an approximate minimum of Equation 4.5. The exact minimum can be obtained by replacing the Equation 4.1 for determining the BMU by

$$c = \arg \min_{1 \le i \le M} \sum_{j} h_{ij} \|\mathbf{x} - \mathbf{m}_i\|^2, \tag{4.6}$$

which is computationally considerably heavier than the winner search of the basic SOM [34].

Maps of process data

In this thesis, measurement data acquired from industrial processes are used as input data for SOMs. Careful preparation of the data is always necessary in order to obtain satisfactory results using any pattern recognition algorithm, and the SOM is not an exception. The data quality issues were already considered in Chapter 3. However, some additional remarks concerning missing values, noise, and outliers with the SOM are commented below.

Missing values. It is usual that a data set contains invalid measurement values (see Section 3.2), which causes problems for any pattern recognition algorithm. The simplest solution to overcome the problem is to discard all the data vectors with one or more such values, but this approach is not always sensible, because it also discards valid data. However, it has been demonstrated that it is sensible to use the available data vector value to find the BMU and to update it with its neighbors [91].

Noise. Physical measurements are corrupted by noise, which is usually assumed to be additive and have the Gaussian distribution. As each model vector of the SOM is a weighted average of the data vectors in the Voronoi set of the map unit and the neighboring map units, noise is reduced and does not usually constitute a major problem.

Outliers. Often, the process data contains some outliers, i.e., individual data vectors which are significantly different from the main body of the data. These vectors strongly contribute to the model vector values in certain parts of the map and are quite easy to detect using visualizations of the SOM discussed in the next section. Alternatively, some other outlier detection technique based on the SOM [79] can also be used. Once the outliers have been found, it is often sensible to remove them from the data set and train the map again in order to obtain a more accurate representation of the main body of the data.

How to use the maps trained with process data

The use of the SOM in process monitoring was already considered in Section 2.1.4 and in the analysis of process data in Section 2.2. Figure 4.1 illustrates these two ways to utilize the SOM with process data. In both cases, a data set is first used to train a map. The vertical direction denotes analysis of the data set the map was trained with; it can be further divided into visualization and clustering, which are described in more detail in Sections 4.1.2 and 4.1.3 below. Clustering can be used as a basis for summarization, i.e., obtaining quantitative information like rules on different regions on the map. The horizontal direction in Figure 4.1 means using the trained map as a reference model to new data. In this context the SOM can either be used to check if the map "knows" a new sample, i.e., novelty detection or to display the map unit and/or set of units which give the best response for the new sample, see Section 4.1.4. The response can naturally be displayed on any visualization of the whole map.

Demonstration data set. The SOM-based methods presented below are demonstrated using a five-dimensional time series with 2780 points from a real pulping process[2], see Figure 4.2. The time series actually consists of three separate regions in time which are all shown in the figure. All the regions describe quality problems of the process, that is, large variation of a output quality variable, the kappa number. The four other variables were in Publication 3 found to be useful in the analysis of the quality variations.

In brief, the goal of the pulping process is delignification, removal of lignin from wood. The kappa number is a quantity which measures the amount of lignin remaining in the pulp after cooking in the digester. There is always a target value for the kappa, which in this case is 32. If the kappa number of the pulp is higher than desired, further processing of the pulp after the cooking becomes difficult. On the other hand, if the kappa number is too small, yield of the process decreases – and in this case, so does runnability of the process.

The SOM-based visualizations presented below played a major role in a research project where sudden drops in the kappa number of a continuous pulp digester were analyzed. In the beginning of the project, the number of variables considered was

[2]The same data were earlier used by the author in [37].

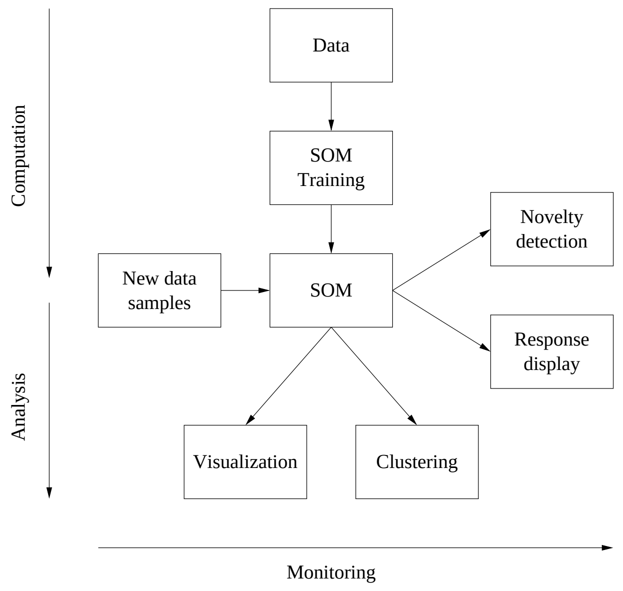


Figure 4.1: Two different ways to utilize the SOM with process data. The vertical direction denotes analysis of a given data set. In the horizontal direction, the SOM is used as a reference for new data.

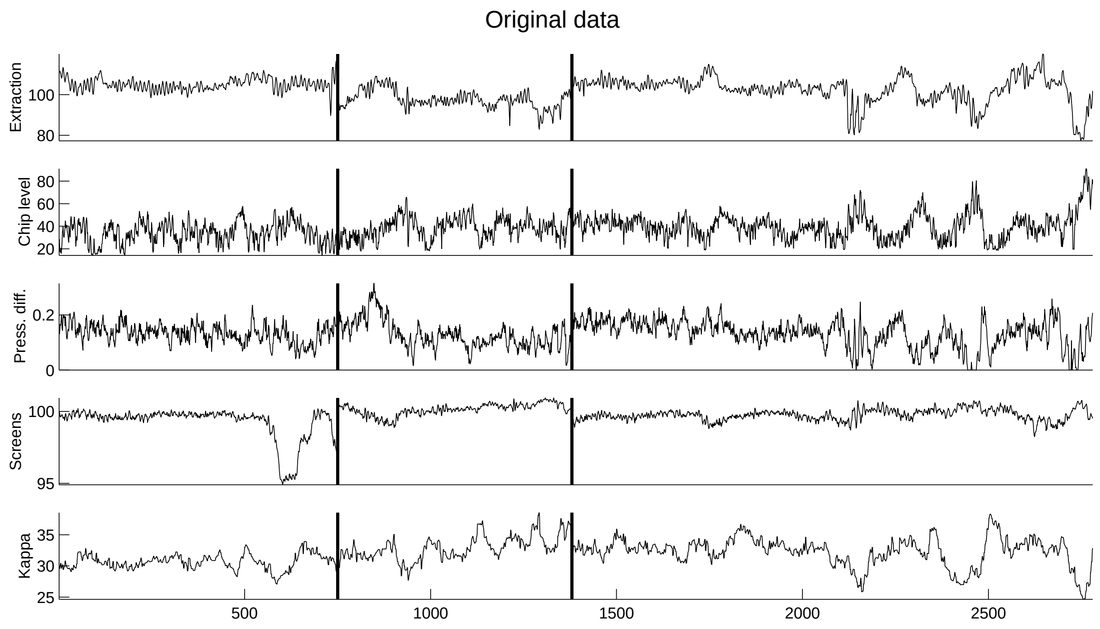


Figure 4.2: A five-dimensional time series which is used to demonstrate the SOM-based methods.

large, say, some dozens. They were gradually reduced down to the most important ones using exploratory data analysis and *a priori* knowledge of the process. Finally, it was possible to give one possible explanation for the observations: the usually smooth vertical movement of the wood chip plug down the digester seemed to slow down in the faulty situations. This resulted in overcooking of the wood chips, that is, decrease of the kappa number. Further, the overcooked pulp choke extraction screens of the digester which slowed down the downward movement of the chip plug even more and so on. Validity of the results was evaluated using a physical process model constructed by a process expert. The model confirmed that the sensitivity of the digester for faults of the assumed type was indeed high [3].

4.1.2 Visualization of the SOM

Perhaps the most important property of the SOM is that it is an efficient method for visualization of high-dimensional data. The SOM is thus an excellent tool in exploratory data analysis, for reviews see [50, 107]. For descriptions and references to SOM-based visualization methods, see for example [37, 39, 106]. The visualizations which are mainly used in this thesis are described in detail below.

Component plane representation displays the values of each model vector element, i.e. each variable, on the map grid. In Figure 4.3, the five component planes of a SOM trained using the data in Figure 4.2 are shown. By inspecting all the component planes simultaneously, one may possibly observe relationships between variables and even roughly distinguish structure of the input data [73]. For instance, consider the illustration of Figure 4.3: one can observe that in the top left corner the value of variable *Kappa* is low, and in the same part of the map the value of variable *Chip level* is high, correspondingly. However, as stated in [66], one has to verify the results carefully in order to avoid misleading conclusions due to effects of vector quantization carried out by the SOM.

Ordering of the component planes [108] makes it easier to investigate a large number of component planes simultaneously. The basic idea is to arrange the component planes in such a way that similar planes (that is, interrelated variables) lie close to each other; this organization is also carried out using the SOM algorithm. In [92], the approach was demonstrated using the pulping data considered in Publication 3.

In **color coding**, each map unit is assigned a color of its own. In Publication 3, the HSV color system (see for example [95]) was used in such a way that the value of component H (hue) was dependent on direction from the center of the map, S (saturation) was held constant, and V (value) was inversely proportional to the map unit distance from the center of the map. Based on the color coding, the following two visualizations can be linked to the component plane representation of the SOM trained using the demonstration data (Figure 4.3).

- Two selected components of the model vectors of the SOM are sketched in a *scatter plot*. As each point in the plot is dyed using the color of the corresponding map unit, it is possible to represent dependence between the two variables in different parts of the map [35]. Top row of Figure 4.4 shows an example of such scatter plots using the demonstration data. All the four other variables are plotted against the variable of interest, the kappa number. It can be detected that in the problematic states of the process – coded by purple color – all the other variables obtain either small or large values. In order to verify that the SOM has properly captured the shape of the data cloud, it is recommended also to plot the original data points in a similar manner and dye each point using the color of the corresponding BMU, see bottom row of Figure 4.4.

- Color coding of the map units can also be used in *time series visualization* in a similar manner. The BMU for each data vector is determined, and the corresponding point in the time series is dyed using the color of the corresponding BMU. This visualization is illustrated in Figure 4.5.

In related works [36, 53], coloring of the SOM was also used with the difference that the changes in the colors between neighboring units were chosen to reflect cluster structure of the model vectors of the map.

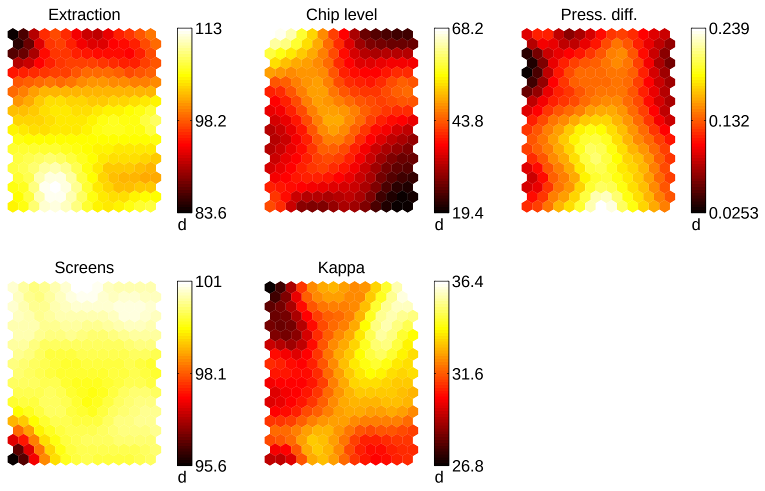


Figure 4.3: Five component planes of the SOM trained using the demonstration data.

4.1.3 Clustering of the SOM

If the number of the model vectors of the SOM is large, but still much less than the number of the data vectors, the model vectors can be treated as a reduced data set which reflects the properties of the underlying data. In Publication 4, clustering of data using the SOM as an intermediate step of cluster analysis was suggested. For an illustration of the approach, see Figure 4.6.

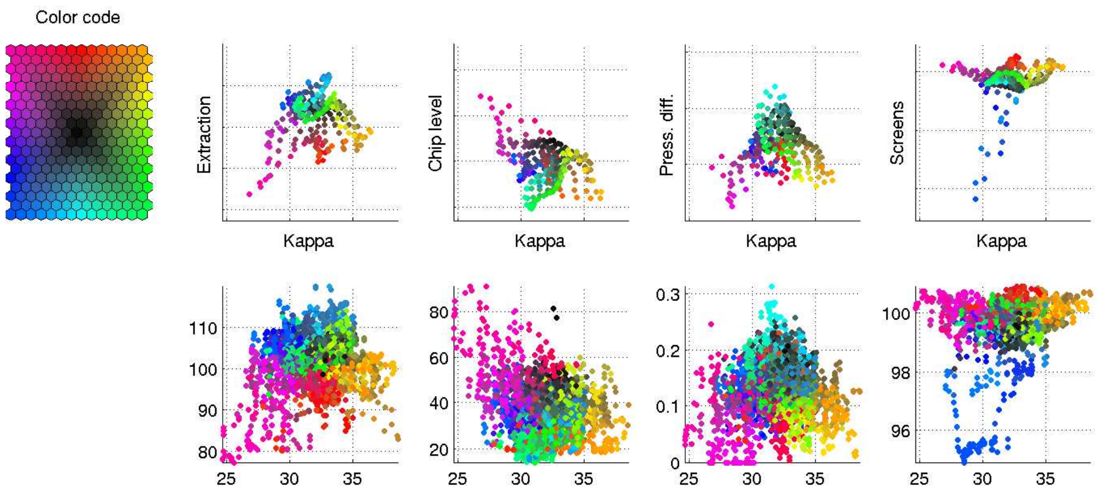


Figure 4.4: Coloring of scatter plots using the SOM. Colors of the map units are shown in the top left corner. The top row also contains scatter plots of four other model vector components against the kappa number. The bottom row consists of similar plots using the original data where each point is dyed using the color of the corresponding BMU.

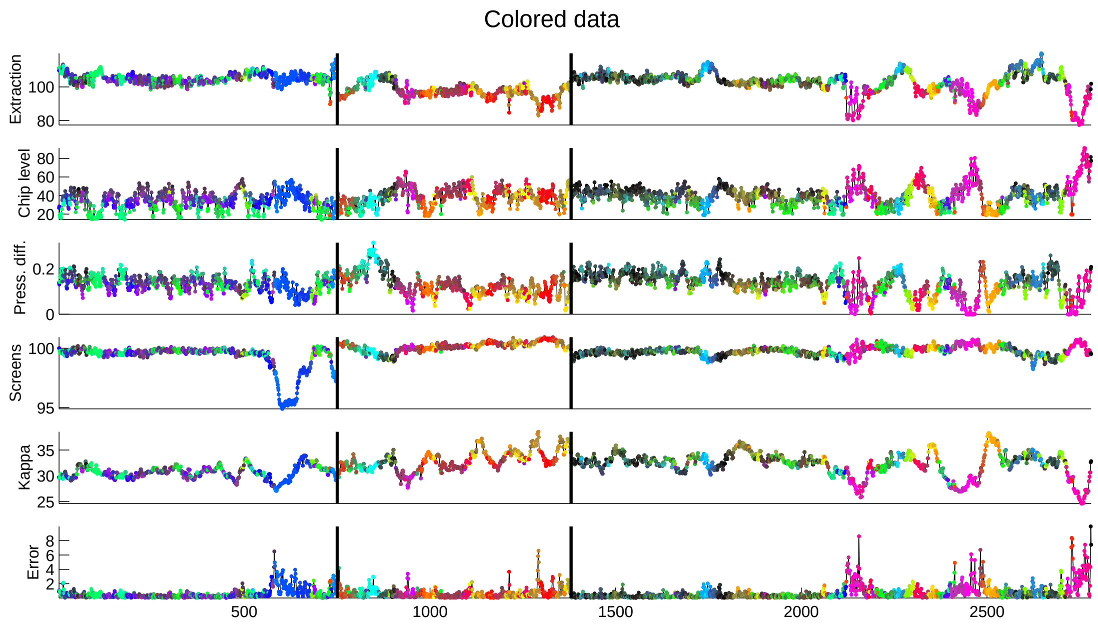


Figure 4.5: Coloring of the time series data using the SOM. All points of the demonstration data have been dyed using the color of the corresponding BMU. The last signal is the quantization error, i.e., the distance between each data point and corresponding BMU. The distance describes how well each data vector is represented by the SOM.

Clustering of the model vectors of the SOM has been considered earlier in [6, 16, 105]. However, the important contribution of Publication 4 is that scalability and accuracy of classic *k*-means and agglomerative clustering algorithms in clustering of the model vectors of the SOM were experimentally tested. Based on the tests of the study, the following conclusions can be made. (1) The approach significantly improves scalability of the clustering algorithms whose computational complexity is $O(N^2)$, that is, the agglomerative algorithms. (2) As the clustering of the model vectors of the SOM was compared to direct clustering of the data, the results were almost identical. Further, the SOM has advantages when the data contains noise and outliers. Because the SOM averages the data, it reduces the effect of the noise and outliers on the clustering result. Also, visualization of the SOM makes it possible to isolate clearly distinguishable outliers from the data.

The clustered SOM can be utilized in two different ways. Either it can be used to obtain quantitative information on the underlying data set in the form of rules [86, 94] or it can be combined with some other visualization techniques to improve readability of the maps. In Figure 4.7, the SOM trained using the demonstration data has been divided into $1 \ldots 10$ clusters using a divisive algorithm (outlined later in Section 4.2.3).

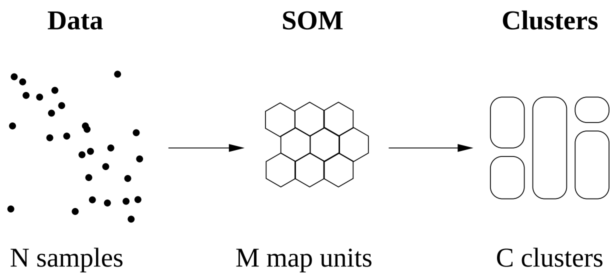


Figure 4.6: Clustering of the data using the SOM. First, a SOM is trained using the data to be clustered. The map units of the SOM are then clustered. Class label of each original data vector is the label of the corresponding BMU.

4.1.4 The SOM as a reference for new data samples

Novelty detection

The goal of novelty detection is to determine whether a before unseen data vector originates from a data distribution which is used as reference. However, the distribution of the reference data is in practice usually unknown, and it needs to be approximated using available data. One possibility is to construct a parametric model based on the vectors of the reference data set. The SOM can be used as such a model of the data.

In novelty detection using the SOM, the simplest possibility is to use the distance between the data vector and the corresponding BMU: if the distance is large, the sam-

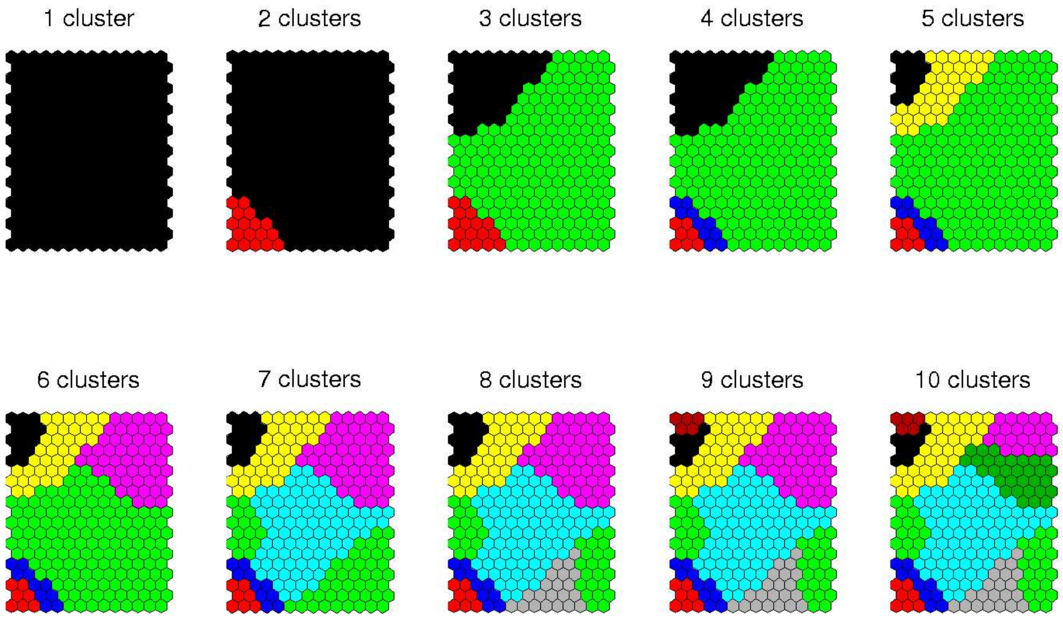


Figure 4.7: Clustering of the SOM into 1 . . . 10 separate classes using a divisive clustering algorithm. In each plot, the class memberships of map units have been indicated using colors.

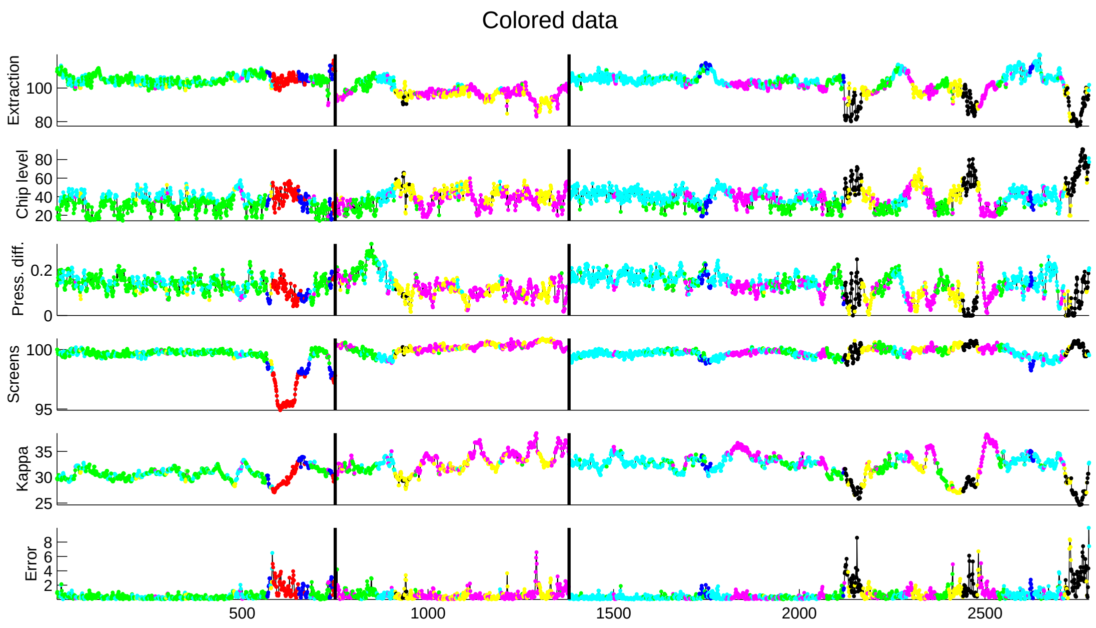


Figure 4.8: Time series plots of the demonstration data. Each data point has been dyed using the color of the corresponding BMU. In this case, the SOM has been divided into seven clusters, see Figure 4.7.

ple is not likely to come from the same data distribution which was used to train the SOM. In [5] and [40], it was suggested that the SOM can also be used as a basis for a mixture model – even though it is not a generative model like, for instance, the Gaussian mixture model (see for example [75]) or the generative topographic mapping [13]. However, in a recent study, a generative probability density model for the SOM was proposed [66].

Responses

The inverse of distance between a data vector and a model vector of a SOM is called *response*. In many applications, indication of the model vector with the best response is necessary, because that model vector is most likely to represent similar data vectors in the input space. For example, in the WEBSOM [62], this methodology is used to retrieve text documents which are similar to the one which was given to the system as input. Also, a mixture model which has been built on top of the SOM can be used [1]. In this case it is possible to compute the probability for each map unit to have generated the input data vector.

In order to illustrate how responses and novelty detection are related to process monitoring, consider Figure 4.9 which contains 200 new data vectors from the same process which was used as a source of the demonstration data. For each of these data points, the corresponding BMU is determined.

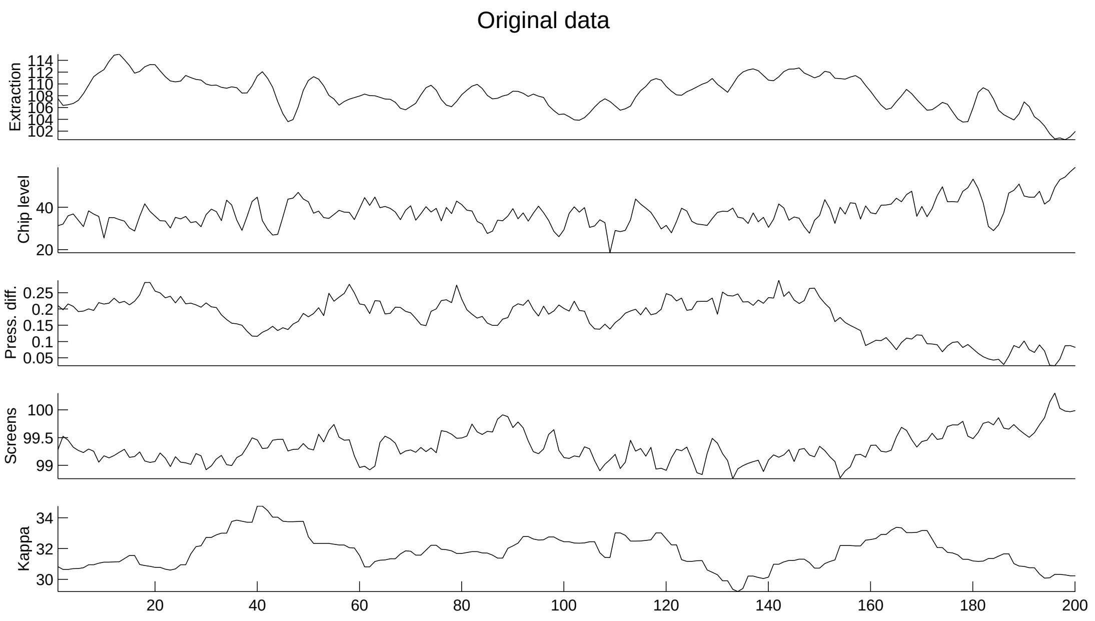


Figure 4.9: A new five-dimensional time series of before unseen data which is used to demonstrate the SOM-based process monitoring.

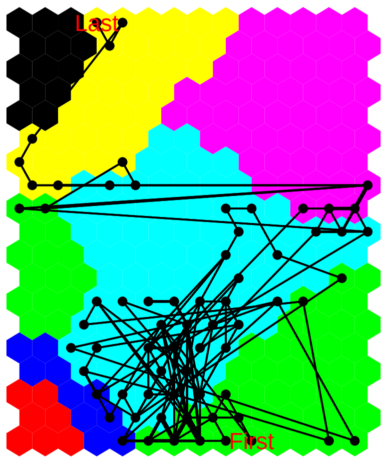


Figure 4.10: A trajectory of responses for new data, which consists of 200 data vectors. The BMUs of the first and the last data vector of the time series have been indicated in the figure.

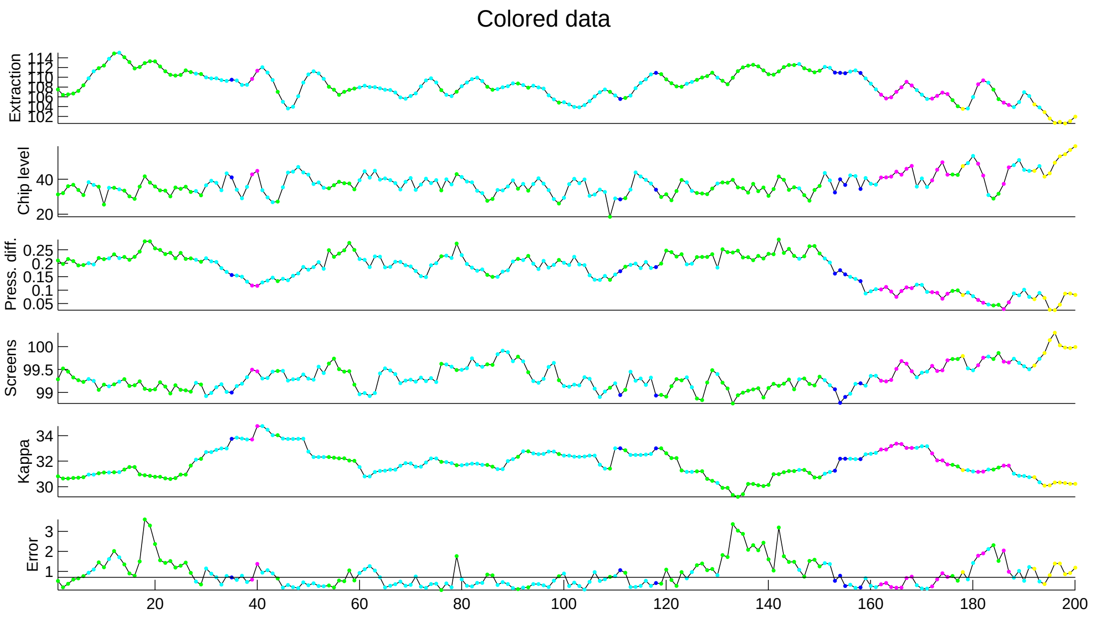


Figure 4.11: Time series plot of the new data with data points dyed according to the colors of the BMUs. The bottom plot shows the quantization error at each time instant.

In Figure 4.10, the SOM trained using the demonstration data set is shown. The map units are assigned colors according to clustering of the SOM to seven clusters (see Figure 4.7). On top of the map, a *trajectory* which passes through all the BMUs of each new data vector is shown. The BMU of each vector has been marked using black dot and adjacent BMUs in time are interconnected using a black line. The trajectory makes it possible to visually indicate the current state of the process and also observe how that state has been reached. An enhancement of the trajectory indication was proposed in [19], where kernel regression was used to transform the discrete SOM grid to a mapping with continuous state indication.

However, location of the BMU does not necessarily properly indicate the current process state, because the new data vector may not be from the same data distribution which was used to train the reference map. Therefore, it is sensible simultaneously to perform novelty detection, which is in this case carried out using quantization error. Figure 4.11 contains time series plots of the signals using the color of the corresponding BMU. The last plot also shows the quantization error dyed in the similar manner. The black horizontal line denotes the average quantization error of the *training data*. In other words, if the error exceeds the line, the quantization error is higher than it was in the training data on the average.

4.2 Cluster analysis

The objective of the cluster analysis is to divide a set of data vectors into groups (or clusters) so that the degree of similarity is strong between vectors of the same cluster and weak between vectors in different clusters. Similarity can be defined in various ways, see for example [7]. In this thesis the Euclidean distance, which is the most commonly used such a measure, is used.

In this section, cluster analysis of multivariate process data is considered. The approach is best suited for data which have been recorded during steady and stable operation of the process, because clustering is only capable of recognizing combinations of measurement levels, not dynamic phenomena. Therefore, the data regions describing changes in the process state, i.e., transients as well as outliers should be removed from the data before clustering.

In Figure 4.12, the phases of cluster analysis of process data are outlined. The cluster analysis begins with feature extraction by segmentation of the data. The segments which contain transient signals and outliers are manually removed from the data. Next, the centroids of the remaining segments are clustered. If interpretation of the clustering results reveals that some of the obtained clusters represent completely uninteresting or even erroneous regions of data, these clusters are removed and the data are clustered again. Below, the steps of the clustering procedure shown in Figure 4.12 are considered in more detail.

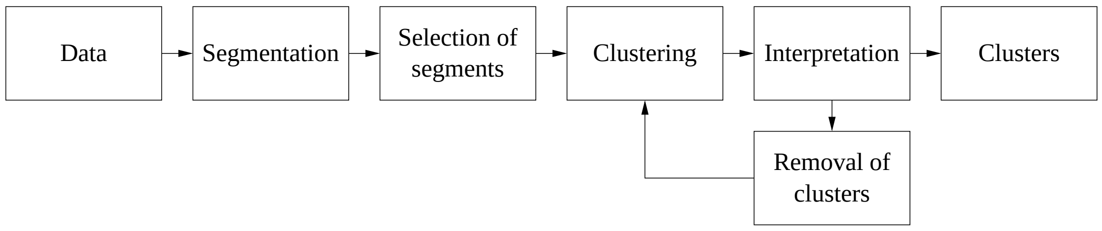


Figure 4.12: The phases of cluster analysis of process data. The segmentation phase can also be seen as a feature extraction phase.

Demonstration data set

Roughly speaking, purpose of a paper machine is to produce paper from mass which consists of water and raw material of the paper like pulp. Removal of major part of the water in different parts of the paper machine is a central task in paper-making. The water removal is carried out in several different ways using suctions, pressing, and evaporation. Large amount of the water is removed in the so-called wet end of the paper machine where the paper mass is fed into the machine. The water removal in the wet end is carried out by several different suctions, partly controlled by the operators of the paper machine.

The cluster analysis scheme presented below was primarily developed to be used in analysis of control actions carried out by operators of a paper machine. Because the operators manually control many central variables – like the suctions of the wet end – it was considered interesting to study if paper with equal quality could be produced using several different settings. If there were many ways, the next task would naturally to be to choose the most cost-effective one.

For demonstration purposes, a 12-dimensional time series which describes control actions of the wet end of a paper machine is used. The data set consists of 6979 data vectors, which are shown in Figure 4.13. The second variable from the top is a nominal variable describing the product grade produced by the paper machine, which is ignored in the segmentation and computation of the clusters, but retained in the data and can thus be used in the interpretation of the results. The following analysis of the demonstration data did not reveal any new information on the process, but is included to illustrate how analysis of a real data set is carried out.

4.2.1 Feature extraction by segmentation

Successive samples in a time series are usually serially correlated. In other words, adjacent samples in time are typically quite similar. Hence, a little essential variation of the data is expected to be lost if the data are segmented by extracting the regions where the time series varies only weakly and representing these segments by their cen-

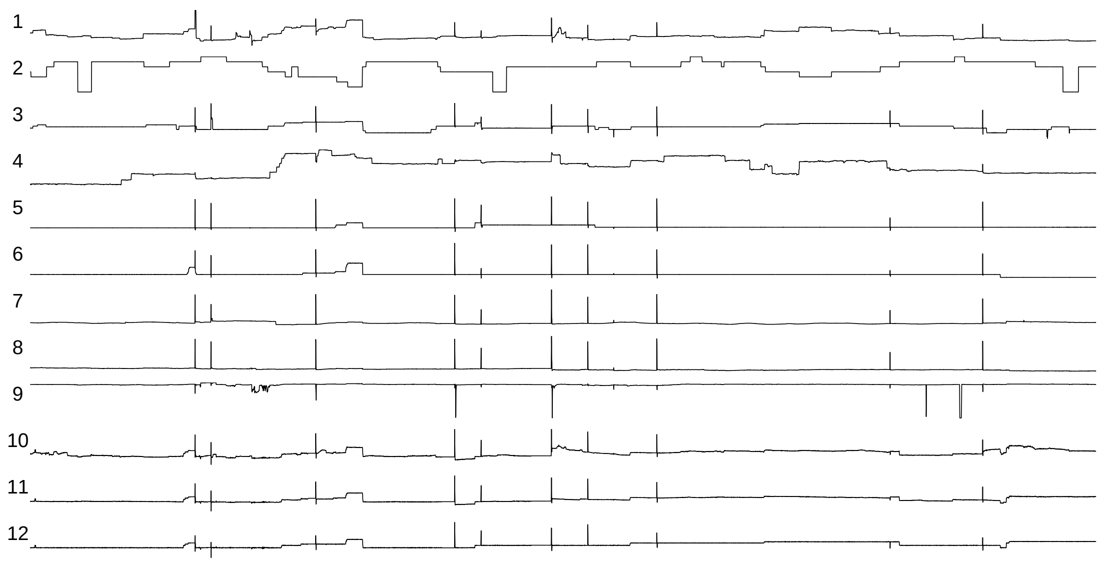


Figure 4.13: Signals used in the demonstration of the cluster analysis of process data.

troids.[3] When these centroids are clustered instead of all the points in the multidimensional time series, the computational load of the clustering algorithms is significantly reduced. Also, the amount of noise in the data is reduced by averaging.

The data could be very efficiently segmented using segments with constant length, but for structured process data, it is more sensible to divide the data into homogeneous regions whose length may vary. An optimal division of multivariate time series with N samples into K segments using dynamic programming [11] is computationally infeasible in large applications, because its computational complexity is $O(KN^2)$. Several feasible approximations with complexity $O(KN)$ can be found in the literature, see for example [27, 33, 57]; these are reviewed in [4]. However, in order to facilitate efficient analysis, the segmentation procedure has to be very fast and an even faster algorithm, discussed in [85], is used.

The algorithm divides a multidimensional time series into segments with a common upper bound E_{ub} for the sum of squared errors (SSE) of each segment. The SSE of a segment that starts at t_0 and ends at t_f is given by[4]

$$E(t_0,t_f) = \sum_{i=t_0}^{t_f} \sum_{j=1}^{d} \left(x_j(i) - \frac{1}{t_f - t_0 + 1} \sum_{k=t_0}^{t_f} x_j(k) \right)^2 . \tag{4.7}$$

Below, the algorithm has been reformulated so that given K, the desired number of segments, it computes an upper bound E_{ub} for the SSE that gives (approximately) K

[3]In Publication 6, vector quantization was used for this task. However, it was later discovered that it is computationally more sensible to use small time segments of the multidimensional time series instead.

[4]In [4] it was shown how a linear line fi t of an arbitrary segment can be computed in unit time after one pass through the data. It is very straightforward to use the same principle here.

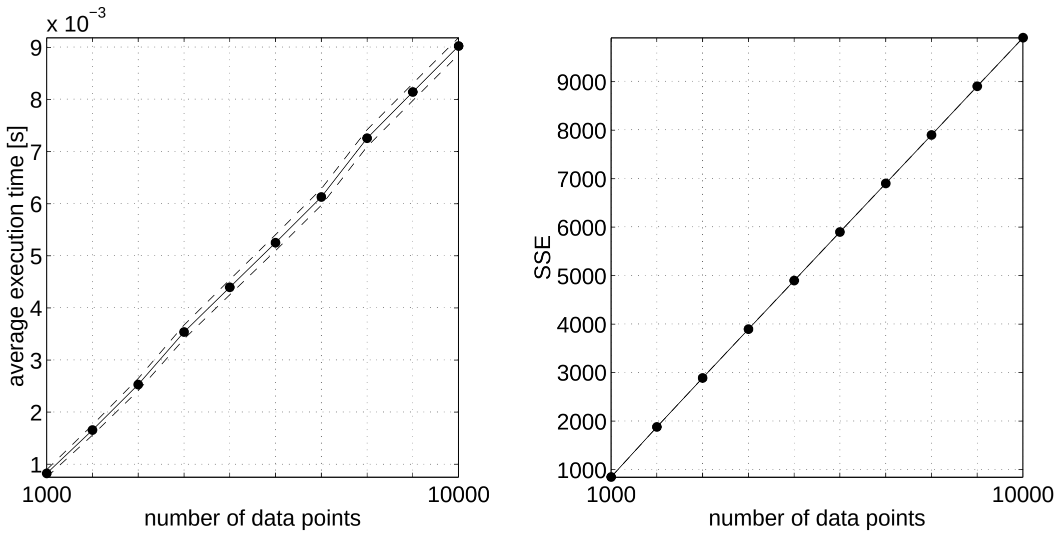
segments. In brief, the algorithm first finds an initial upper bound (the lower bound is naturally 0) and uses the bisection search algorithm (see for example [10, pp. 276–277]) to find the one that gives the desired number of segments.

Outline of the segmentation algorithm.

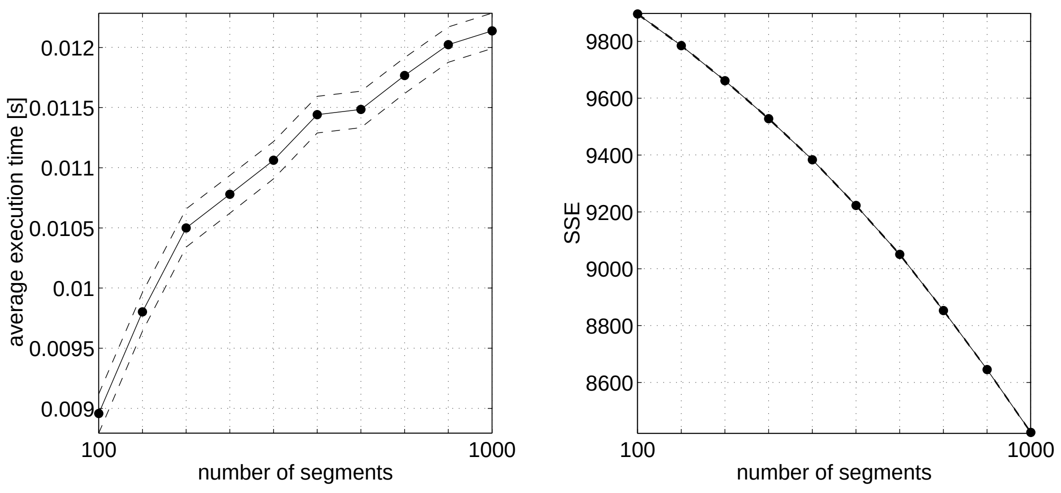
1. Specify K, the desired number of segments. Also choose ε, a distinguishability constant.
2. The lower bound of the error, $E_{\mathrm{lb}} = 0$. The initial value of the upper bound is given by $E_{\mathrm{ub}} = \max_{1 \le i \le N-w+1} \{E(i, i+w-1)\}$, where $w = \lceil \frac{N}{K} \rceil$.
3. Set $E_{\mathrm{ub}} = E_{\mathrm{lb}} + (E_{\mathrm{ub}} - E_{\mathrm{lb}})/2 = E_{\mathrm{ub}}/2$ and step size $\Delta = (E_{\mathrm{ub}} - E_{\mathrm{lb}})/4$.
4. (a) Set the number of the current segment $k = 1$, the number of the current sample $t = 1$, and the number of the first sample in the current segment $t_0 = 1$.
   (b) Compute $E(t_0, t)$, the SSE of the current segment; if $E(t_0, t) > E_{\mathrm{ub}}$, let $n_k = t - t_0$, $k = k + 1$, and $t_0 = t$.
   (c) Sample $\mathbf{x}(t)$ belongs to segment k.
   (d) Let $t = t + 1$. If $t > N$, quit; otherwise go back to (b).
5. If $k = K$ or $\Delta < \varepsilon$, quit. The upper bound is E_{ub}.
6. If $k > K$, let $E_{\mathrm{ub}} = E_{\mathrm{ub}} + \Delta$; otherwise, let $E_{\mathrm{ub}} = E_{\mathrm{ub}} - \Delta$.
7. Let $\Delta = \Delta/2$ and go to step 4.

As the primary goal of the segmentation is reduction of the data, it is sensible to formulate the algorithm in this manner. It is difficult to find a reasonable value for E_{ub}, but it is usually known how many segments can be used to keep the execution time of a clustering algorithm feasible.

Because the computational complexity of the algorithm is not easy to analyze, it was evaluated experimentally. Figure 4.14 contains results of tests where the performance of the segmentation algorithm was tested using random data. In Figure 4.14(a), the number of segments was 100 in all runs whereas in Figure 4.14(b), the number of data points was 10000 in all cases. The results suggest that the computational complexity of the algorithm is linear with respect to the number of data points but merely logarithmic with respect to the number of segments.



(a)



(b)

Figure 4.14: Performance curves of the segmentation algorithm on page 44. (a) Execution time (left) and SSE (right) when the number of data points is varied between 1000 and 10000; the number segments is 100 in all cases. (b) Execution time (left) and SSE (right) when the number of segments is varied between 100 and 1000; the number of data points is 10000 in all cases. The solid line depicts the mean over 10000 runs and the dashed line is the standard deviation of the mean.

4.2.2 Selection of the segments

As the segmentation algorithm produces segments with error that has an upper bound which is common for all segments, the transients and the segments with outliers consist of fewer samples than the steady state segments. This fact can be used in derivation of a heuristical measure to guide the analyst to manually determine the segments which should be removed. Denote the SSE of each segment by E_i, and the number of data vectors in the ith segment by n_i, respectively. Using these two quantities, the following heuristical quantity can be computed for each segment:

$$s_i = n_i + \left(1 - \frac{E_i}{\max_{1 \le i \le K}\{E_i\}}\right), \tag{4.8}$$

where s_i describes tendency of ith segment to be a transient or an outlier segment. The second term is significant only if segments of equal length are removed one by one: it guarantees that segments with large errors (which are more likely to be a transient or outlier segment) are removed first.

The next task is to determine a threshold s_{lim} so that all segments for which $s_i < s_{\text{lim}}$ are removed from the data. Since automatic selection of such a threshold is difficult, it should be selected manually by the analyst. The decision is supported by interactive computer tools, which guide the analyst to select an appropriate value for s_{lim}. In Figures 4.15(a)–(d), visualizations provided by such a tool are shown. Each visualization is briefly commented below.

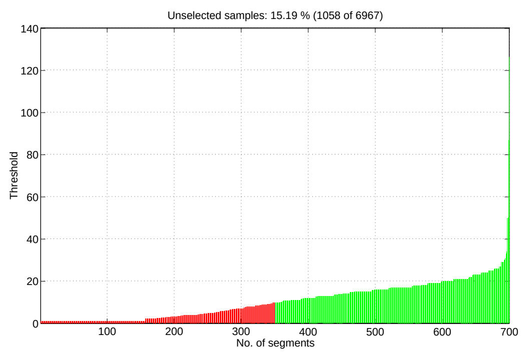
- Figure 4.15(a) shows the current value of s_i for all segments of the demonstration data sorted in ascending order. This view only tells the amount of data that is to be thrown away/retained.

- Figure 4.15(b) illustrates the selected and unselected data points of one time series. Also, it shows which segment is next removed if s_{lim} is increased. Before clustering it is sensible to browse through all the variables using such a time series display to ensure that proper segments have been removed.

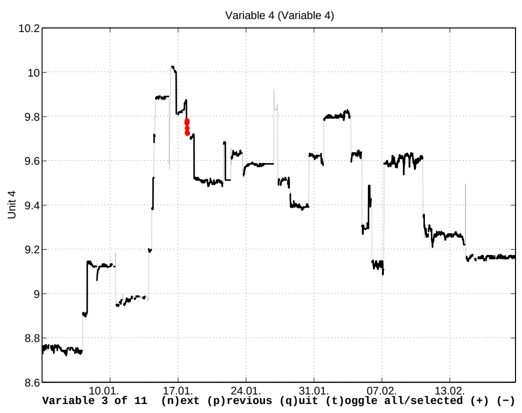
- In Figure 4.15(c), the centroids of the selected segments are projected in three two-dimensional subspaces spanned by the three first principal components (PCs). Using this display, it is possible to observe some centroids far away from the main body of the data.[5] For example, a group of outlying centroids can be observed in the direction of the third PC.

- The variables which are most likely to include the outlying values have the greatest absolute coefficient values in the third PC. Projection of the whole data on the third PC and illustration of coefficients of the third PC is shown in Figure 4.15(d). The variable with the greatest variance in the third PC is thus the variable number one.

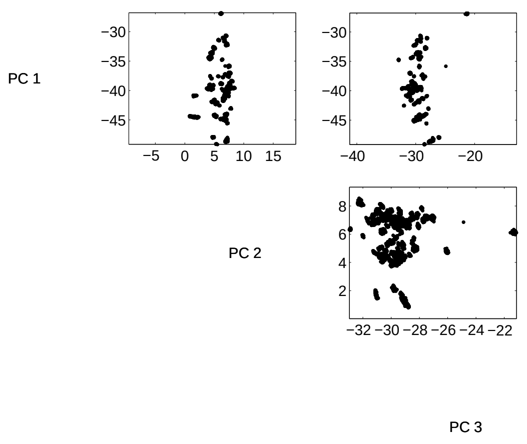
[5]Note that some outliers may not be visible in these projections, but may be detected using projection on some other principal components.



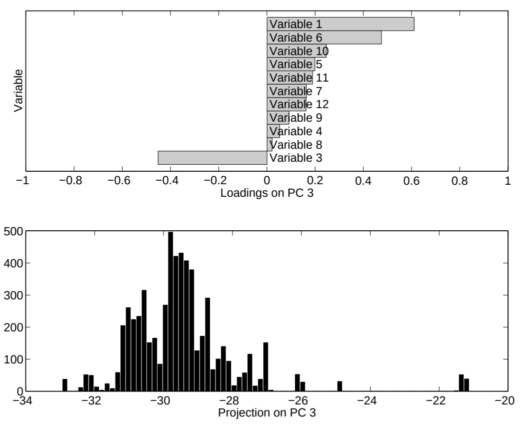
(a)



(b)



(c)



(d)

Figure 4.15: Illustration of the displays which support the selection of the threshold s_{lim}. (a) Value of s_i for all segments of the demonstration data sorted in ascending order. The segments which would be discarded using the current value of s_{lim} are shown using red color. (b) Selected (black color) and unselected data points (gray color) of the variable number four. The red data points belong to the segment which is next removed if s_{lim} is increased. (c) Projections of the data in three different two-dimensional subspaces spanned by the three first PCs. (d) Projection of the data on the third PC and the coefficients of each variable in the third PC.

4.2.3 Clustering algorithms

A clustering algorithm assigns given data vectors into clusters according to some criterion, which either explicitly or implicitly defines a cluster. In the literature, a large number of clustering algorithms has been reported, see e.g. [7, 23, 47, 55]. Because the algorithms differ from each other in several aspects, it is difficult to construct a taxonomy for them. In the following, however, some ways to distinguish between different types of algorithms are listed.

Partitive vs. hierarchical. In partitive clustering, the number of the clusters usually is fixed in advance, but may also be determined by the clustering algorithm, see for example [14]. The hierarchical algorithms usually build a binary clustering tree, a *dendrogram*, which can be cut at any level to obtain a desired number of clusters. The hierarchical algorithms can be further divided into agglomerative and divisive. The former ones initially assign all data points into clusters of their own and build the dendrogram by combining two clusters at each step. The latter algorithms start with all points in a single cluster which is first split into two subclusters that are at the next step again split until there is only one sample left in each cluster.

Crisp vs. soft memberships. In the crisp algorithms, one sample belongs to exactly one group whereas in the clustering algorithms with soft memberships, a sample may belong to many clusters with varying degree of membership. The degree of membership can be used to measure uncertainty in assignment of a sample in each class. There exists two ways to express this uncertainty: probability and fuzzy membership, which lead to mixture model [75] and fuzzy clustering algorithms [21].

Parametric vs. non-parametric representation. A group may have a parametric representation, which is in the simplest case cluster centroid, or in a more complex case, e.g., centroid and a covariance matrix. The well-known *k*-means algorithm is of the former type and the Gaussian mixture model is of the latter type. In non-parametric approach, the cluster is only defined by its samples.

The clustering results reported in Publication 6 were carried out using a divisive hierarchical algorithm [72], which starts with all data vectors in a single cluster. At each step, the cluster with the largest diameter is split into two subclusters as suggested in [55]. The diameter is defined by maximum distance between any two points in the same cluster.

Outline of the divisive algorithm.

1. Set the number of clusters $l = 1$; set all samples to cluster X_l and set cluster $X_{l+1} = \varnothing$.

2. For each sample $\mathbf{x}_i$ of X_l, compute the average distance to all the other samples of X_l:
$$d(\mathbf{x}_i, X_l \setminus \{i\}) = \frac{1}{n_l - 1} \sum_{j \in X_l,\, j \neq i} d(\mathbf{x}_i, \mathbf{x}_j).$$
Denote the sample that maximizes the dissimilarity by $\mathbf{x}_{i'}$.

3. Move the vector $\mathbf{x}_{i'}$ from cluster X_l to cluster X_{l+1}:
$$\begin{aligned} X_l &= X_l \setminus \{i'\} \\ X_{l+1} &= X_{l+1} \cup \{i'\} \end{aligned}$$

4. Find the $i' \in X_l$ that maximizes
$$d(\mathbf{x}_{i'}, X_l \setminus \{i'\}) - d(\mathbf{x}_{i'}, X_{l+1}) = \frac{1}{n_l - 1} \sum_{j \in X_l, j \neq i'} d(\mathbf{x}_{i'}, \mathbf{x}_j) - \frac{1}{n_{l+1}} \sum_{k \in X_{l+1}} d(\mathbf{x}_{i'}, \mathbf{x}_k)$$

5. If $d(\mathbf{x}_{i'}, X_l \setminus \{i'\}) - d(\mathbf{x}_{i'}, X_{l+1}) > 0$, go to step 3 else go to step 6.

6. If all the clusters have only one sample, stop. Otherwise, find the cluster X_m with the largest diameter:
$$\mathrm{diam}(X_m) = \max_{j \in X_m, k \in X_m} d(\mathbf{x}_j, \mathbf{x}_k),$$
set $l = l + 1$, $X_l = X_m$ and $X_{l+1} = \varnothing$ and go to step 2.

The algorithm outlined above is hierarchical with crisp memberships and non-parametric cluster representation. A clear disadvantage is that the computational complexity of the algorithm is high, quadratic with respect to the number of samples. On the other hand, the computational burden can be considerably reduced by clustering the segment centroids instead of the original data. In analysis of real process data, the divisive algorithm outlined above has proved to be very useful; its use can be motivated by the following arguments.

- **Hierarchical nature.** Determination of the number of clusters has been extensively studied in the pattern recognition literature (see e.g. [76]). However, real data in practice usually has no clear and unique cluster structure. Therefore, it is sensible to interactively explore the data and try different number of clusters. A hierarchical algorithm is especially suitable for this purpose since all partitionings can be obtained by cutting the dendrogram at a suitable level.

- **Isolation of outliers.** The algorithm is sensitive to outliers, which is often an undesired property. However, this can be benefited in pruning of the clustering result, because the outliers often constitute clusters of their own. Once recognized, they can be easily discarded and the data can then be clustered again. However, if the outliers are extracted from the data before any other clusters, there is no need to recluster the data – which is also a very attractive property of the algorithm.

- **Consistent results.** The algorithm is deterministic, i.e., it always produces the same clustering tree for the same data in each run. However, it should be acknowledged that if the data changes slightly, the clustering result may also change.

4.2.4 Interpretation

After the clustering, each data vector is assigned the label (or identifier or class) of the cluster it belongs to. However, as the clustering has been carried out in an unsupervised manner, it is not known what kind of data each of the obtained clusters represents. Therefore one crucial step in applications of cluster analysis is the interpretation of the results [47]. In interpretation of the clustering results of process data, the following questions are typically sought answers for.

- What is the number of data points in each cluster?
- How are the clusters located in time?
- How can the clusters be characterized: which variables best describe each cluster and how?
- What are the differences and similarities between clusters?
- Which variables are the most significant in discrimination between clusters?

Answering the two first questions in the list above is simple and straightforward: an illustrative display is shown in Figure 4.16. The three last questions in the list above can be answered when a quantity measuring the ability of a variable to discriminate between two multivariate data partitions is derived. It is obvious that there are many possible ways to select such a measure. For example, if the clusters can be assumed to be Gaussian, classic standard methods like Fisher discriminant analysis (see for example [15, pp. 57–59]), can be used.

In Publication 5, another common such a measure, the Kolmogorov-Smirnov (K-S) statistic (see for example [98, pp. 1187–1192]) was used. Use of the K-S statistic can be motivated by the fact that it is simple, invariant to linear scaling of data, it does not make assumptions on distribution of the data, and it is not especially sensitive to

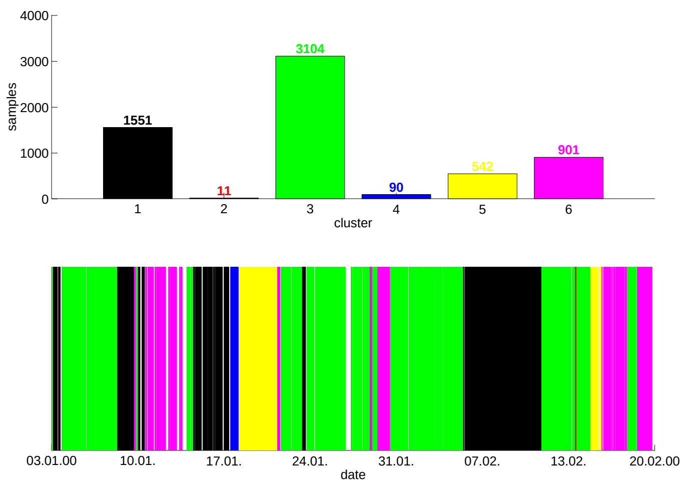


Figure 4.16: The number of samples in each cluster (top) and the location of each cluster in time (bottom) where the example data has been divided into six clusters. In both illustrations, different clusters have been indicated using colors.

outliers. Because many clustering algorithms (like the divisive algorithm used in this thesis) do not assume any data distribution, use of a such a measure can be motivated.

The K-S statistic is defined as the maximum absolute difference between two cumulative distribution functions, say, $F_{k,i}(l)$ and $F_{k,j}(l)$:

$$D_k(X_i, X_j) = \max_{-\infty < l < \infty} |F_{k,i}(l) - F_{k,j}(l)|, \tag{4.9}$$

where k denotes variable number and i and j are the indices of the partitions which are compared. If the functions $F_{k,i}(l)$ and $F_{k,j}(l)$ are unknown, as they are in this case, they can be estimated using data in a very straightforward manner. Figure 4.17 illustrates the use of the K-S statistic between two data distributions.

Characterization of a partition. Denote the partition of interest by X_i. First, the K-S statistics $D_k(X_i, X_j)$ for each variable k is computed by comparing the variable distributions in the partition X_i with the other partitions $X_{\text{others}} = \bigcup_{j, j \neq i} X_j$. The K-S statistic $D_k(X_i, X_{\text{others}})$ reflects the importance of variable k in discrimination between partition X_i and all the other partitions: the greater the value of the K-S statistic, the more important variable k is in characterization of partition X_i.

In Figure 4.18, a display of one cluster of the example data is shown. The names of the variables are shown in the order of significance from top to bottom in the left panel. To the right, there is a “flat histogram” display of the cluster. For each variable, all the points which do not belong to the cluster are drawn using gray color and the points which are members of the cluster by black color. It can be observed that nearly all variables get consistently high values in this cluster. However, it is sensible to also look at the detailed presentations for each variable, see Figure 4.19.

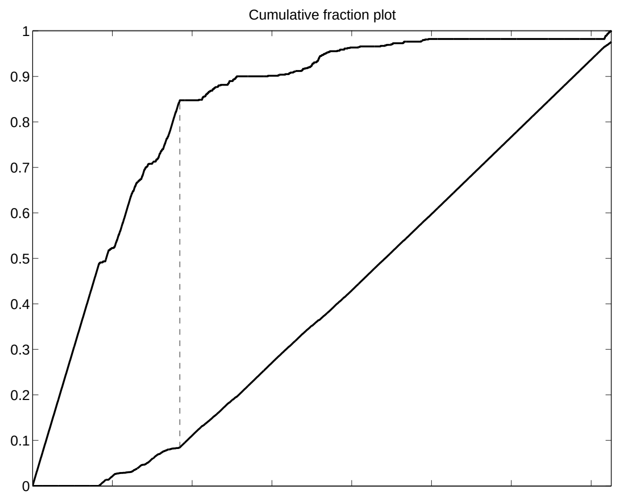


Figure 4.17: An example of use of the K-S statistic. The K-S statistic between two data distributions is the maximum difference between the corresponding cumulative data distributions. The maximum difference is shown by a dotted line.

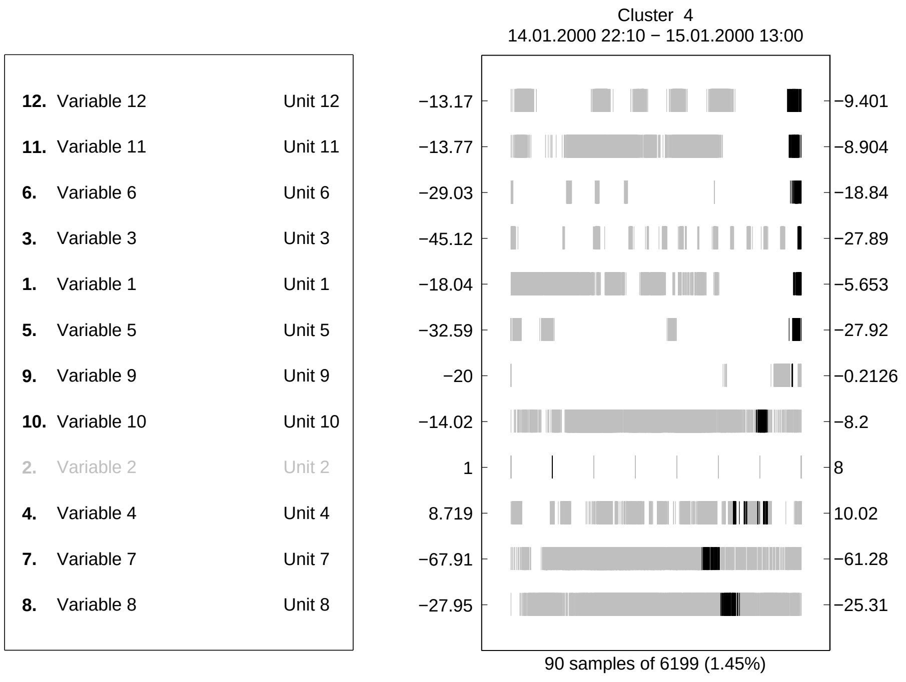


Figure 4.18: A display of cluster number four. The variables are sorted in order of significance from top to bottom. Names of the variables are shown in the left panel and contents the variables in the right panel, correspondingly. In the right panel, all the points which belong to cluster four are drawn using black and the points that belong to other clusters using gray color, respectively.

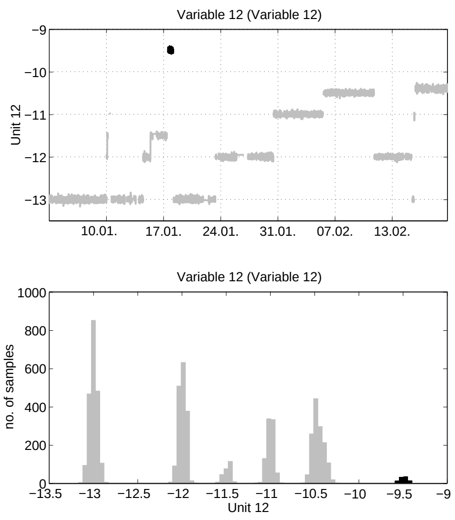


Figure 4.19: A focused view of variable 12 which best explains the cluster number four: time series plot (top) and histogram (bottom). The coloring is similar to Figure 4.18.

Comparison of two partitions. In comparison of two partitions X_i and X_j, the K-S statistics $D_k(X_i, X_j)$ are computed for each variable k by comparing partition X_i with X_j. Now the value of $D_k(X_i, X_j)$ reflects the ability of the variable k to discriminate between partitions X_i and X_j.

In Figure 4.20, a cluster comparison display between two clusters is shown. The display is quite similar to the display of one cluster in Figure 4.18 with the difference that now the order of the variables has been determined according to their ability to discriminate between the two clusters.

Discrimination ability of variables. The last item in the list given in the beginning of this section is determination of the variable(s) which best discriminate between the clusters. The discrimination ability of kth variable can be computed using the K-S statistic in the following way:

$$D_k^{\text{tot}} = \sum_{i=1}^{d-1} \sum_{j=i+1}^{d} D_k(X_i, X_j), \tag{4.10}$$

where $D_k(X_i, X_j)$ is the K-S statistic between the kth variable in the clusters X_i and X_j. The variable with the greatest cluster discrimination ability is the one for which the value of D_k^{tot} is greatest. In the case of the example data the variable is number three, which is shown in Figure 4.21; all the points of the figure have dyed been using the color of the corresponding cluster.

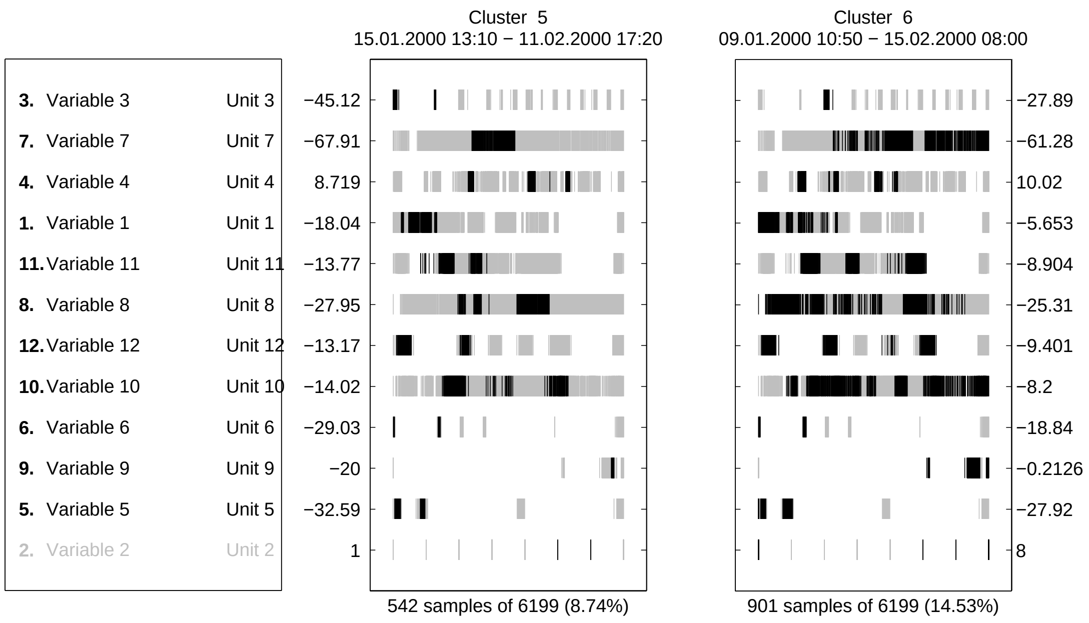


Figure 4.20: A display of comparison between the clusters five and six. The variables are sorted in order of significance from top to bottom. Variable names are shown in the left display. Contents of each variable in cluster five are shown in the middle panel and contents of cluster six in the right panel, correspondingly. All the points which belong to the clusters five and six have been drawn using black and the points that belong to other clusters using gray color in the panels corresponding to the clusters.

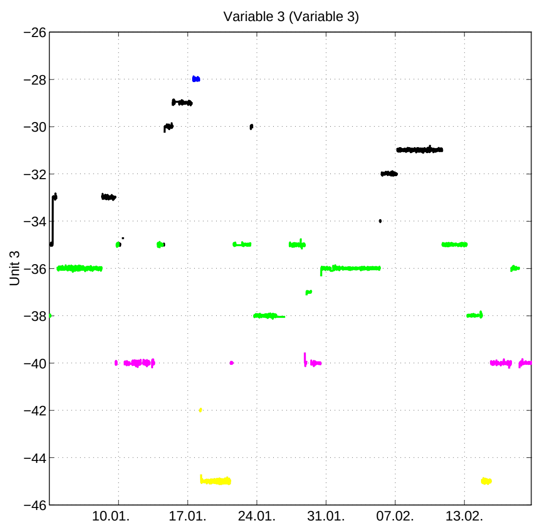


Figure 4.21: Time series plot of variable number three which best separates between the clusters. Each point in the plot is dyed using the color of the corresponding cluster. As it can be observed, the variable is almost alone able to separate between the six clusters.

Chapter 5

Conclusions

The main contribution of this thesis is that both the SOM and the cluster analysis have been applied in novel ways to real-world problems to explore real process data of large-scale industrial processes. The studies have concretely confirmed the well-known fact that dealing with real-world problems unavoidably requires large amount of background work: specification of the problem, acquisition and preparation of the data, and interpretation of the results with process experts. The work carried out in this thesis has contributed to the computation of the unsupervised models and, especially, their interpretation by visualization.

Exploratory data analysis cannot be carried by a person who is not familiar with the analysis methods, because there are many technical details that unavoidably need to be known in order to achieve sensible results. All the phases in the analysis can never be automated, but the tasks which require human interaction or interpretation should be supported by computer-based tools as efficiently as possible. This has been one of the main goals in this thesis: to take some steps toward more user-friendly and efficient exploratory analysis of process data.

The SOM is a powerful method for display of high-dimensional data in two dimensions. If the actual dimension of the data is higher than two, the map is often capable of making a reasonable projection, but the quality of the projection weakens as the dimension of the data grows. Because determination of the actual data dimension is in practice very difficult, validation of the mapping carried out by the SOM is not straightforward. Fortunately, some measures (see for example [52]) for the quality of the maps are available. They do not directly measure actual dimension of the data, but are useful in validation as well as comparison of maps.

For a person who is not familiar with the SOM, the map displays are usually at first confusing. For example, it takes time to realize why the axes of the two-dimensional map do not always have some simple interpretation. However, after the basic idea of the SOM has been adopted, the SOM is often found to be a useful visualization method which makes it possible to obtain an illustrative display of high-dimensional data.

The cluster analysis alone or combined with the SOM can be used to study the structure of the data. All the clustering algorithms contain either implicit or explicit definition of a cluster, but there is no guarantee that the definition agrees with anything that is found in real data. Many clustering algorithms assume that data contain clusters of regular shape like sphere or ellipsoid. Fortunately, even though the structure of real data is often more complicated, some structure can be discovered in many cases using the simple assumptions.

In the future work in the field of analysis of process data, the following two things should be considered in depth.

- In the studies presented in this thesis, unsupervised methods were used to explore and visualize properties of multivariate data, which is indeed very important. However, after the exploratory phase the findings and the generated hypotheses need to be validated. Statistical validation of the results has not been considered in this thesis, but should be studied in the future work.

- The methods used in this thesis produce static models, which are suitable for analysis of steady state operation of a process. In that case, even though the process would be dynamic, it can be assumed static. Modeling of process dynamics using the SOM is considered in [42], and a review on representations of time in model based on the SOM can be found in [18]. If the dynamics of the process should also be included in the models used in this thesis, guidelines can be found in these two studies.

References

[1] J. Ahola, E. Alhoniemi, and O. Simula. Monitoring industrial processes using the self-organizing map. In *Proceedings of the 1999 IEEE Midnight-Sun Workshop on Soft Computing Methods in Industrial Applications*, pages 22–27, June 1999.

[2] J. T. Alander, M. Frisk, L. Holmström, A. Hämäläinen, and J. Tuominen. Process error detection using self-organizing feature maps. In T. Kohonen, K. M. kisa ra, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, volume 2, pages 1229–1232. Elsevier, 1991.

[3] E. Alhoniemi. *Process measurement based analysis of continuous kraft pulping process*. Licentiate's Thesis, Helsinki University of Technology, 1998. In Finnish.

[4] E. Alhoniemi. Simplified time series representations for efficient analysis of industrial process data. *Artificial Intelligence in Engineering Design, Analysis and Manufacturing*, 2002. Submitted.

[5] E. Alhoniemi, J. Himberg, and J. Vesanto. Probabilistic measures for responses of self-organizing map units. In H. Bothe, E. Oja, E. Massad, and C. Haefke, editors, *Proceedings of the International ICSC Congress on Computational Intelligence Methods and Applications*, pages 286–290. ICSC Academic Press, 1999.

[6] C. Ambroise, G. Sèze, F. Badran, and S. Thiria. Hierarchical clustering of self-organizing maps for cloud classification. *Neurocomputing*, 30:47–52, 2002.

[7] M. R. Anderberg. *Cluster Analysis for Applications*. Academic Press, 1973.

[8] C. Apté. Data mining: An industrial research perspective. *IEEE Computational Science and Engineering*, 4(2):6–9, April-June 1997.

[9] M. Basseville. Detecting changes in signals and systems – a survey. *Automatica*, 24(3):309–326, 1988.

[10] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear programming: theory and algorithms*. John Wiley & Sons, 1993.

[11] R. Bellman. On the approximation of curves by line segments using dynamic programming. *Communications of the ACM*, 4(6):284, June 1961.

[12] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[13] C. M. Bishop, M. Svensén, and C. K. I. Williams. GTM: The generative topographic mapping. *Neural Computation*, 10(1):215–235, 1998.

[14] P. Cheeseman and J. Stutz. Bayesian classification (AutoClass): Theory and results. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 151–180. AAAI Press/MIT Press, 1996.

[15] L. H. Chiang, E. L. Russell, and R. D. Braaz. *Fault Detection and Diagnosis in Industrial Systems*. Springer, 2001.

[16] M. Cottrell, P. Gaubert, P. Letremy, and P. Rousset. Analyzing and representing multidimensional quantitative and qualitative data: Demographic study of the rhône valley. the domestic consumption of the canadian families. In E. Oja and S. Kaski, editors, *Kohonen maps*, pages 1–14. Elsevier, 1999.

[17] C. M. Crowe. Data reconciliation – progress and challenges. *Journal of Process Control*, 6(2–3):89–98, 1996.

[18] G. de A. Barreto and A. F. R. Araújo. Time in self-organizing maps: An overview of models. *International Journal of Computer Research*, 10(2):139–179, 2001.

[19] I. Díaz, A. B. Diez, and A. A. C. Vega. Complex process visualization through continuous feature maps using radial basis functions. In G. Dorffner, H. Bischof, and K. Hornik, editors, *Artificial Neural Networks – ICANN 2001*, number 2130 in Lecture Notes in Computer Science, pages 443–449. Springer, 2001.

[20] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Sons, 2nd edition, 2001.

[21] D. Dumitrescu, B. Lazzerini, and L. C. Jain. *Fuzzy sets and their application to clustering and training*. CRC Press, 2000.

[22] E. Erwin, K. Obermayer, and K. Schulten. Self-organizing maps: Ordering, convergence properties and energy functions. *Biological Cybernetics*, 67(1):47–55, 1992.

[23] B. S. Everitt. *Cluster Analysis*. Arnold, 3rd edition, 1993.

[24] P. M. Frank. Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy – a survey and some new results. *Automatica*, 3(26):459–474, 1990.

[25] H. Furukawa, T. Ueda, and M. Kitamura. A systematic method for rational definition of plant diagnostic symptoms by self-organizing neural networks. *Neurocomputing*, 13:171–183, 1996.

[26] E. Govekar, E. Susič, P. Mužič, and I. Grabec. Self-organizing neural network application to technical process parameter estimation. In I. Alexander and J. Taylor, editors, *Artificial Neural Networks, 2*, volume 1, pages 579–582. Elsevier, 1992.

[27] V. Guralnik and J. Srivastava. Event detection from time series data. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 33–42, 1999.

[28] F. Gustafsson. *Adaptive Filtering and Change Detection*. John Wiley & Sons, 2000.

[29] J. F. Hair, Jr., R. E. Anderson, R. L. Tatham, and W. C. Black. *Multivariate data analysis*. Prentice-Hall, 5th edition, 1998.

[30] D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. MIT Press, 2001.

[31] T. Harris. A Kohonen S.O.M. based, machine health monitoring system which enables diagnosis of faults not seen in the training set. In *Proceedings of the 1993 International Joint Conference on Neural Networks*, volume 1, pages 947–950, 1993.

[32] T. Harris, L. Gamlyn, P. Smith, J. MacIntyre, A. Brason, R. Palmer, H. Smith, and A. Slater. 'NEURAL-MAINE': Intelligent on-line multiple sensor diagnostics for steam turbines in power generation. In *Proceedings of the 1995 IEEE International Conference on Neural Networks*, volume 2, pages 686–691, 1995.

[33] D. M. Hawkins. Point estimation of the parameters of piecewise regression models. *Applied Statistics*, 25(1):51–57, 1976.

[34] T. M. Heskes and B. Kappen. Error potential for self-organization. In *Proc. ICNN'93, International Conference on Neural Networks*, volume III, pages 1219–1223, Piscataway, NJ, 1993. IEEE Service Center.

[35] J. Himberg. Enhancing SOM-based data visualization by linking different data projections. In L. Xu, L. W. Chan, I. King, and A. Fu, editors, *Intelligent data engineering and learning*, pages 427–434. Springer, 1998.

[36] J. Himberg. A SOM based cluster visualization and its application for false coloring. In *Proceedings of International Joint Conference on Neural Networks*, volume 3, pages 587–592, 2000.

[37] J. Himberg, J. Ahola, E. Alhoniemi, J. Vesanto, and O. Simula. The self-organizing map as a tool in knowledge engineering. In N. R. Pal, editor, *Pattern recognition in soft computing paradigm*, pages 38–65. World Scientific, 2001.

[38] D. M. Himmelblau. *Fault Detection and Diagnosis in Chemical and Petrochemical Processes*. Elsevier, 1978.

[39] E. Häkkinen and P. Koikkalainen. SOM based visualization in data analysis. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, editors, *Proceedings of the 7th International Conference on Artificial Neural Networks*, number 1327 in Lecture Notes in Computer Science, pages 601–606. Springer, 1997.

[40] A. Hämäläinen. *Self-organizing Map and Reduced Kernel Density Estimation*. PhD thesis, University of Helsinki, 1995.

[41] A. J. Hoffman and N. T. van der Merve. The application of neural networks to vibrational diagnostics for multiple fault conditions. *Computer Standards & Interfaces*, 24:139–149, 2002.

[42] H. Hyötyniemi. *Self-Organizing Artificial Neural Networks in Dynamic Systems Modeling and Control*. PhD thesis, Helsinki University of Technology, 1994.

[43] J. Iivarinen, K. Heikkinen, J. Rauhamaa, P. Vuorimaa, and A. Visa. A defect detection scheme for web surface inspection. *International Journal of Pattern Recognition and Artificial Intelligence*, 14(6):735–755, 2000.

[44] R. Isermann. Process fault detection based on modeling and estimation methods – a survey. *Automatica*, 20(4):387–404, 1984.

[45] R. Isermann. Supervision, fault detection and fault-diagnosis methods – an introduction. *Control Engineering Practice*, 5(5):639–652, May 1997.

[46] R. Isermann and P. Ballé. Trends in the application of model-based fault detection and diagnosis of technical processes. *Control Engineering Practice*, 5(5):709–719, May 1997.

[47] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[48] S.-L. Jämsä-Jounela, M. Vermasvuori, S. Haavisto, and J. Kämpe. Industrial applications of the intelligent fault diagnosis system. In *Proceedings of the American Control Conference*, volume 6, pages 4437–4442, 2001.

[49] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986.

[50] S. Kaski. *Data Exploration Using Self-Organizing Maps*. PhD thesis, Helsinki University of Technology, Feb. 1997.

[51] S. Kaski, J. Kangas, and T. Kohonen. Bibliography of self-organizing map (SOM) papers: 1981–1997. *Neural Computing Surveys*, 1:102–350, 1998.

[52] S. Kaski and K. Lagus. Comparing self-organizing maps. In C. von der Malsburg, W. von Seelen, J. C. Vorbruggen, and B. Sendhoff, editors, *Proceedings of International Conference on Artificial Neural Networks*, volume 1112 of *Lecture Notes in Computer Science*, pages 809–814. Springer, 1996.

[53] S. Kaski, J. Venna, and T. Kohonen. Coloring that reveals cluster structures in multivariate data. *Australian Journal of Intelligent Information Processing Systems*, 6:82–88, 2000.

[54] M. Kasslin, J. Kangas, and O. Simula. Process state monitoring using self-organizing maps. In I. Aleksander and J. Taylor, editors, *Artificial Neural Networks, 2*, volume 2, pages 1531–1534. Elsevier, 1992.

[55] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[56] KCL Development. *Wedge 5.0 User's manual I: Basic usage*, 2001. In Finnish.

[57] E. J. Keogh and M. J. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, pages 239–243, 1998.

[58] T. Kohonen. Self-organizing maps: optimization approaches. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, volume II, pages 981–990, Amsterdam, Netherlands, 1991. North-Holland.

[59] T. Kohonen. *Self-Organizing Maps*. Springer, 3rd edition, 1997.

[60] T. Kohonen. The self-organizing map. *Neurocomputing*, 21:1–6, 1998.

[61] T. Kohonen, J. Hynninen, J. Kangas, and J. Laaksonen. SOM_PAK: The self-organizing map program package. Technical Report A31, Helsinki University of Technology, Laboratory of Computer and Information Science, 1996.

[62] T. Kohonen, S. Kaski, K. Lagus, J. Salojärvi, J. Honkela, V. Paatero, and A. Saarela. Self organization of a massive document collection. *IEEE Transactions on Neural Networks*, 11(3):574–585, May 2000.

[63] T. Kohonen, E. Oja, O. Simula, A. Visa, and J. Kangas. Engineering applications of the self-organizing map. *Proceedings of the IEEE*, 84(10):1358–84, 1996.

[64] J. T. Laaksonen, J. M. Koskela, S. P. Laakso, and E. Oja. PicSOM – content-based image retrieval with self-organizing maps. *Pattern Recognition Letters*, 21(13–14):1199–1207, Dec. 2000.

[65] S. Laine, K. Pulkkinen, and S.-L. Jämsä-Jounela. On-line determination of the concentrator feed type at Outokumpu Hitura mine. *Minerals Engineering*, 12(8–9):881–895, 2000.

[66] J. Lampinen and T. Kostiainen. Generative probability density model in the self-organizing map. In U. Seiffert and L. Jain, editors, *Recent advances in self-organizing neural networks*, pages 75–94. Physica Verlag, 2002.

[67] J. Lampinen and O. Taipale. Optimization and simulation of quality properties in paper machine with neural networks. In *IEEE International Conference on Neural Networks*, volume 6, pages 3812–3815, 1994.

[68] J. R. Leigh. *Applied digital control: theory, design, and implementation*. Prentice-Hall, 2nd edition, 1992.

[69] L. Ljung. *System Identification: Theory for the User*. Prentice-Hall, 1987.

[70] W. L. Luyben. *Process Modeling, Simulation, and Control for Chemical engineers*. McGraw-Hill, 2nd edition, 1990.

[71] J. F. MacGregor and T. Kourti. Statistical process control of multivariate processes. *Control Engineering Practice*, 3(3):403–414, Mar. 1995.

[72] P. Macnaughton-Smith, W. T. Williams, M. B. Dale, and L. G. Mockett. Dissimilarity analysis: a new technique of hierarchical sub-division. *Nature*, 202:1034–1035, 1964.

[73] K. M. Marks and K. F. Goser. Analysis of VLSI process data based on self-organizing feature maps. In *Proceedings of International Workshop on Neural Networks & Their Applications (Neuro-Nimes '88)*, pages 327–348, 1988.

[74] T. E. Marlin. *Process control: designing processes and control systems for dynamic performance*. McGraw-Hill, 2nd edition, 2000.

[75] G. J. McLahlan and K. E. Basford. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, 1988.

[76] G. W. Milligan and M. C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, June 1985.

[77] D. W. Moolman, C. Aldrich, G. P. J. Schmitz, and J. S. J. van Deventer. The interrelationship between surface froth characteristics and industrial flotation performance. *Minerals Engineering*, 9(8):837–854, 1996.

[78] H. Mori, Y. Tamaru, and S. Tsuzuki. An artificial neural-net based technique for power system dynamic stability with the Kohonen model. *IEEE Transactions on Power Systems*, 7(2):856–864, 1992.

[79] A. Muñoz and J. Muruzábal. Self-organizing maps for outlier detection. *Neurocomputing*, pages 33–60, 1998.

[80] D. Niebur and A. J. Germond. Power system static security assessment using the Kohonen neural network classifier. *IEEE Transactions on Power Systems*, 7(2):865–872, 1992.

[81] J. P. Norton. *An Introduction to Identification*. Academic Press, 1988.

[82] J. S. Oakland. *Statistical Process Control*. Butterworth-Heinemann, 4th edition, 1999.

[83] T. R. Padmanabhan. *Industrial Instrumentation*. Springer, 2000.

[84] R. J. Patton, F. J. Uppal, and C. J. Lopez-Toribio. Soft computing approaches to fault diagnosis for dynamic systems: A survey. In *Proceeding of the 4th IFAC Symposium on Fault Detection, Supervision, and Safety for Technical Processes*, volume 1, pages 298–311, 2000.

[85] T. Pavlidis. Waveform segmentation through functional approximation. *IEEE Transactions on Computers*, C-22(7):689–697, July 1973.

[86] W. Pedrycz and H. C. Card. Linguistic interpretation of self-organizing maps. In *IEEE International Conference on Fuzzy Systems*, pages 371–378. IEEE, 1992.

[87] K. Raivio, O. Simula, and J. Laiho. Neural analysis of mobile radio access network. In *Proceedings of the International Conference on Data Mining (ICDM)*, pages 457–464, 2001.

[88] J. T. Rantanen, S. J. Laine, O. K. A. J.-P. Mannermaa, O. E. Simula, and J. K. Yliruusi. Visualization of fluid-bed granulation with self-organizing maps. *Journal of Pharmaceutical and Biomedical Analysis*, 24:343–352, 2001.

[89] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[90] J. A. Romagnoli and M. C. Sánchez. *Data Processing and Reconciliation for Chemical Process Operations*. Academic Press, 2000.

[91] T. Samad and S. A. Harp. Self-organization with partial data. *Network: Computation in Neural Systems*, 3:205–212, 1992.

[92] O. Simula and E. Alhoniemi. SOM based analysis of pulping process data. In *Proceedings of International Work-Conference on Artificial and Natural Neural Networks*, volume II, pages 567–577. Springer, 1999.

[93] O. Simula and J. Kangas. Process monitoring and visualization using self-organizing maps. In A. B. Bulsari, editor, *Neural Networks for Chemical Engineers*, pages 371–384. Elsevier, 1995.

[94] M. Siponen, J. Vesanto, O. Simula, and P. Vasara. An approach to automated interpretation of SOM. In *Proceedings of workshop on self-organizing maps*, pages 89–94. Springer, 2001.

[95] A. R. Smith. Color gamut transform pairs. *Computer graphics*, 12(3):12–19, Aug. 1978.

[96] T. Sorsa, H. N. Koivo, and H. Koivisto. Neural networks in process fault diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(4):815–825, july/august 1991.

[97] G. Stephanopoulos and C. Han. Intelligent systems in process engineering: a review. *Computers and Chemical Engineering*, 20(6–7):743–791, 1996.

[98] A. Stuart and J. K. Ord. *Kendall's Advanced Theory of Statistics*, volume 2. Edward Arnold, 5th edition, 1991.

[99] E. L. Sutanto and K. Warwick. Multivariable cluster analysis for high-speed industrial machinery. *IEE Proceedings on Science, Measurement and Technology, IEE Proceedings*, 142(5):417–423, Sept. 1995.

[100] P. Teppola, S.-P. Mujunen, and P. Minkkinen. Adaptive fuzzy c-means clustering in process monitoring. *Chemometrics & Intelligent Laboratory Systems*, 45:23–38, 1999.

[101] V. Tryba and K. Goser. Self-organizing feature maps for process control in chemistry. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, volume 1, pages 847–852. Elsevier, 1991.

[102] A. Ultsch. Self organized feature maps for monitoring and knowledge acquisition of a chemical process. In S. Gielen and B. Kappen, editors, *Proceedings of the International Conference on Artificial Neural Networks*, pages 864–867. Springer-Verlag, 1993.

[103] J. S. J. van Deventer, D. W. Moolman, and C. Aldrich. Visualization of plant disturbances using self-organizing feature maps. *Computers and Chemical Engineering*, 20 Supplement(972):S1095–S1100, May 1996.

[104] M. Vapola, O. Simula, T. Kohonen, and P. Meriläinen. Representation and identification of fault conditions of an anaesthesia system by means of the self-organizing map. In M. Marinaro and P. G. Morasso, editors, *Proceedings of the International Conference on Artificial Neural Networks*, volume 1, pages 350–353. Springer-Verlag, 1994.

[105] A. Varfis and C. Versino. Clustering of socio-economic data with Kohonen maps. *Neural Network World*, 2(6):813–834, 1992.

[106] J. Vesanto. SOM-based data visualization methods. *Intelligent Data Analysis*, 3(2):111–126, 1999.

[107] J. Vesanto. *Data Exploration Process Based on the Self-Organizing Map*. PhD thesis, Helsinki University of Technology, May 2002.

[108] J. Vesanto and J. Ahola. Hunting for correlations in data using the self-organizing map. In H. Bothe, E. Oja, E. Massad, and C. Haefke, editors, *Proceedings of the International ICSC Congress on Computational Intelligence Methods and Applications (CIMA '99)*, pages 279–285. ICSC Academic Press, 1999.

[109] J. Vesanto, J. Himberg, E. Alhoniemi, and J. Parhankangas. SOM Toolbox for Matlab 5. Technical Report A57, Helsinki University of Technology, Laboratory of Computer and Information Science, Apr. 2000.

[110] X. Z. Wang. *Data Mining and Knowledge Discovery for Process Monitoring and Control*. Advances in Industrial Control. Springer, 1999.

[111] X. Z. Wang and C. McGreavy. Automatic classification for mining process operational data. *Industrial & Engineering Chemistry Research*, 37(6):2215–2222, June 1998.

[112] A. Ypma and R. P. W. Duin. Novelty detection using self-organizing maps. In N. Kasabov, R. Kozma, K. Ko, R. O'Shea, G. Coghill, and T. Gedeon, editors, *Proceedings of International Conference on Neural Information Processing*, volume 2, pages 1322–1325. Springer, 1997.

[113] J. Zupan, M. Novic, and I. Ruisánchez. Kohonen and counterpropagation artificial neural networks in analytical chemistry. *Chemometrics & Intelligent Laboratory Systems*, 38:1–23, 1997.